

GEM3D

Generated by Doxygen 1.6.1

Thu Nov 30 13:09:51 2017

Contents

1	Main Page	1
2	Directory Hierarchy	3
2.1	Directories	3
3	Namespace Index	5
3.1	Namespace List	5
4	Class Index	7
4.1	Class Hierarchy	7
5	Class Index	9
5.1	Class List	9
6	File Index	11
6.1	File List	11
7	Directory Documentation	13
7.1	/home/jhasbestan/Morton_Parallel_v0/src/include/ Directory Reference	13
7.2	/home/jhasbestan/Morton_Parallel_v0/src/ Directory Reference	14
8	Namespace Documentation	15
8.1	Abstraction Namespace Reference	15
8.1.1	Enumeration Type Documentation	15
8.1.1.1	DataType	15
8.2	stl Namespace Reference	16
8.2.1	Function Documentation	16
8.2.1.1	operator<<	16
8.2.1.2	operator<<	16
8.2.1.3	parse_float	16
8.2.1.4	parse_point	16

8.2.1.5	parse_stl	16
9	Class Documentation	17
9.1	CenterCoords Struct Reference	17
9.1.1	Detailed Description	17
9.1.2	Member Data Documentation	17
9.1.2.1	x	17
9.1.2.2	y	17
9.1.2.3	z	17
9.2	CommCollective< Type > Class Template Reference	18
9.2.1	Detailed Description	18
9.2.2	Constructor & Destructor Documentation	18
9.2.2.1	CommCollective	18
9.2.2.2	CommCollective	18
9.2.2.3	~CommCollective	19
9.2.3	Member Function Documentation	19
9.2.3.1	bcast	19
9.2.3.2	getTotalNumber	19
9.2.3.3	getTotalNumber	19
9.2.3.4	lbcast	19
9.2.3.5	IgetTotalNumber	19
9.2.3.6	IgetTotalNumber	19
9.2.3.7	waitOnRequest	19
9.2.4	Member Data Documentation	19
9.2.4.1	Com	19
9.2.4.2	Msg	19
9.3	CommPoint2Point< Type > Class Template Reference	21
9.3.1	Detailed Description	21
9.3.2	Constructor & Destructor Documentation	21
9.3.2.1	CommPoint2Point	21
9.3.2.2	CommPoint2Point	22
9.3.2.3	CommPoint2Point	22
9.3.2.4	~CommPoint2Point	22
9.3.3	Member Function Documentation	22
9.3.3.1	assignReciever	22
9.3.3.2	assignSender	22
9.3.3.3	getOffset	22

9.3.3.4	Irecv	22
9.3.3.5	Isend	22
9.3.3.6	myRank	23
9.3.3.7	mySize	23
9.3.3.8	recv	23
9.3.3.9	send	23
9.3.4	Member Data Documentation	23
9.3.4.1	Com	23
9.3.4.2	Msg	23
9.4	DataType Class Reference	24
9.4.1	Detailed Description	24
9.5	Forest< N, Nvalue, M, Mvalue > Class Template Reference	25
9.5.1	Detailed Description	26
9.5.2	Constructor & Destructor Documentation	27
9.5.2.1	Forest	27
9.5.2.2	~Forest	27
9.5.3	Member Function Documentation	27
9.5.3.1	assignGeom	27
9.5.3.2	assignSeeds	27
9.5.3.3	checkGraphConsistency	27
9.5.3.4	checkNbrsOfNbrsConsistency	27
9.5.3.5	checkWithNbrs	27
9.5.3.6	checkZoltanPartConsistency	27
9.5.3.7	combinedLevel	27
9.5.3.8	comPatternConstruct	28
9.5.3.9	constructCommWeak	28
9.5.3.10	constructElementKeyForRcvdMessage	28
9.5.3.11	constructSeedKeyForRcvdMessage	28
9.5.3.12	convertBitsToDouble	28
9.5.3.13	convertDoubleToBits	28
9.5.3.14	createCommGraph	28
9.5.3.15	createNbrsOfNbrs	28
9.5.3.16	debug	28
9.5.3.17	debugDerefine	28
9.5.3.18	encodeGeometry	28
9.5.3.19	findFlipLevel	28

9.5.3.20	findSeedLevelForRcvdMessage	29
9.5.3.21	flipAll	29
9.5.3.22	flipForNbr	29
9.5.3.23	forestsize	29
9.5.3.24	fourToOneBalance	29
9.5.3.25	getDirections	29
9.5.3.26	getElemNbrs	29
9.5.3.27	getListEachTree	30
9.5.3.28	getMaxSeedsLevel	30
9.5.3.29	getNbrSeedLevel	30
9.5.3.30	getTotalMeshSize	30
9.5.3.31	getTotalSize	30
9.5.3.32	isInSeed	30
9.5.3.33	moveGeom	30
9.5.3.34	pushToDerefineEachTree	30
9.5.3.35	rcvrMessageSize	30
9.5.3.36	recoverAllZeroSingularity	30
9.5.3.37	refineEachTree	30
9.5.3.38	refineEachTreeVoxel	31
9.5.3.39	refineForestBalanced	31
9.5.3.40	removeAllZeroSingularity	31
9.5.3.41	retainFourToOneBalance	31
9.5.3.42	zoltanGeomrepart	31
9.5.4	Friends And Related Function Documentation	31
9.5.4.1	Phdf5	31
9.5.5	Member Data Documentation	31
9.5.5.1	ancestorcoords	31
9.5.5.2	ancestorlength	31
9.5.5.3	Com	32
9.5.5.4	destination	32
9.5.5.5	geom	32
9.5.5.6	graphComm	32
9.5.5.7	maxseedlevel	32
9.5.5.8	nbrsOfNbrs	32
9.5.5.9	npx	32
9.5.5.10	npy	32

9.5.5.11	npz	32
9.5.5.12	seeds	32
9.5.5.13	trees	32
9.5.5.14	zoltan_out	33
9.5.5.15	zz	33
9.6	FullOctreeTop< N > Class Template Reference	34
9.6.1	Constructor & Destructor Documentation	34
9.6.1.1	FullOctreeTop	34
9.6.1.2	~FullOctreeTop	34
9.6.2	Member Function Documentation	34
9.6.2.1	checkGraphConsistency	34
9.6.2.2	constructNbrProcs	35
9.6.2.3	convertRank2Bits	35
9.6.2.4	isBoundary	35
9.6.2.5	readNbrs	35
9.6.2.6	readRoot	35
9.6.3	Member Data Documentation	35
9.6.3.1	fullOctreeLevel	35
9.6.3.2	Nbr	35
9.6.3.3	rootKey	35
9.7	FullTree< N, value > Class Template Reference	36
9.7.1	Detailed Description	36
9.7.2	Constructor & Destructor Documentation	37
9.7.2.1	FullTree	37
9.7.2.2	~FullTree	37
9.7.3	Member Function Documentation	37
9.7.3.1	assignProcs	37
9.7.3.2	begin	37
9.7.3.3	convertCoordToMorton	37
9.7.3.4	end	37
9.7.3.5	find	37
9.7.3.6	getLevel	37
9.7.3.7	insertKey	37
9.7.3.8	isBoundary	38
9.7.3.9	level	38
9.7.3.10	nbrsConstrcut	38

9.7.3.11	setLevel	38
9.7.3.12	size	38
9.7.4	Member Data Documentation	38
9.7.4.1	derefinelist	38
9.7.4.2	fixedlevel	38
9.7.4.3	mesh	38
9.7.4.4	mortonSTL	39
9.7.4.5	refinelist	39
9.8	GraphData Struct Reference	40
9.8.1	Detailed Description	40
9.8.2	Member Data Documentation	40
9.8.2.1	c	40
9.8.2.2	nbrGID	40
9.8.2.3	nbrIndex	40
9.8.2.4	nbrProc	40
9.8.2.5	numMyVertices	40
9.8.2.6	vertexGID	40
9.9	MeshData Struct Reference	41
9.9.1	Detailed Description	41
9.9.2	Member Data Documentation	41
9.9.2.1	c	41
9.9.2.2	myGlobalIDs	41
9.9.2.3	numGlobalPoints	41
9.9.2.4	numMyPoints	41
9.9.2.5	w	41
9.10	Message Struct Reference	42
9.10.1	Detailed Description	42
9.10.2	Member Function Documentation	42
9.10.2.1	print	42
9.10.3	Member Data Documentation	42
9.10.3.1	buf	42
9.10.3.2	count	42
9.10.3.3	datatype	42
9.10.3.4	reciever	42
9.10.3.5	request	43
9.10.3.6	sender	43

9.10.3.7	status	43
9.10.3.8	tag	43
9.11	MpiCom Struct Reference	44
9.11.1	Detailed Description	44
9.11.2	Constructor & Destructor Documentation	44
9.11.2.1	MpiCom	44
9.11.2.2	MpiCom	44
9.11.3	Member Data Documentation	44
9.11.3.1	comsize	44
9.11.3.2	mpicom	44
9.11.3.3	myrank	44
9.12	Phdf5< N, Nvalue, M, Mvalue > Class Template Reference	45
9.12.1	Detailed Description	45
9.12.2	Constructor & Destructor Documentation	45
9.12.2.1	Phdf5	45
9.12.2.2	~Phdf5	45
9.12.3	Member Function Documentation	45
9.12.3.1	writeMultiBlock	45
9.12.3.2	writePolyvertex	46
9.12.3.3	xdmfMultiBlock	46
9.12.3.4	xdmfPolyvertex	46
9.12.4	Member Data Documentation	46
9.12.4.1	totalnumber	46
9.13	stl::point Struct Reference	47
9.13.1	Detailed Description	47
9.13.2	Constructor & Destructor Documentation	47
9.13.2.1	point	47
9.13.2.2	point	47
9.13.3	Member Data Documentation	47
9.13.3.1	x	47
9.13.3.2	y	47
9.13.3.3	z	47
9.14	stl::stl_data Struct Reference	48
9.14.1	Detailed Description	48
9.14.2	Constructor & Destructor Documentation	48
9.14.2.1	stl_data	48

9.14.3	Member Data Documentation	48
9.14.3.1	name	48
9.14.3.2	triangles	48
9.15	TemplateForest< N, Nvalue, M, Mvalue, T > Class Template Reference	49
9.15.1	Constructor & Destructor Documentation	50
9.15.1.1	TemplateForest	50
9.15.1.2	~TemplateForest	51
9.15.2	Member Function Documentation	51
9.15.2.1	assignGeom	51
9.15.2.2	assignSeeds	51
9.15.2.3	combinedLevel	51
9.15.2.4	comPatternConstruct	51
9.15.2.5	comPatternConstruct	51
9.15.2.6	constructElementKeyForRcvdMessage	51
9.15.2.7	constructSeedKeyForRcvdMessage	51
9.15.2.8	encodeGeometry	51
9.15.2.9	findFlipLevel	51
9.15.2.10	findSeedLevelForRcvdMessage	52
9.15.2.11	findSeedLevelForRcvdMessage	52
9.15.2.12	flipAll	52
9.15.2.13	flipForNbr	52
9.15.2.14	forestsize	52
9.15.2.15	fourToOneBalance	52
9.15.2.16	getDirections	52
9.15.2.17	getElemNbrs	52
9.15.2.18	getListEachTree	53
9.15.2.19	getMaxSeedsLevel	53
9.15.2.20	getNbrSeedLevel	53
9.15.2.21	getNbrSeedLevel	53
9.15.2.22	getTotalMeshSize	53
9.15.2.23	getTotalSize	53
9.15.2.24	isInSeed	53
9.15.2.25	moveGeom	53
9.15.2.26	nonCollectiveNbrComm	53
9.15.2.27	recoverAllZeroSingularity	53
9.15.2.28	refineEachTree	53

9.15.2.29	refineEachTreeVoxel	54
9.15.2.30	refineForestBalanced	54
9.15.2.31	removeAllZeroSingularity	54
9.15.3	Friends And Related Function Documentation	54
9.15.3.1	Phdf5	54
9.15.4	Member Data Documentation	54
9.15.4.1	ancestorcoords	54
9.15.4.2	ancestorlength	54
9.15.4.3	Com	54
9.15.4.4	destination	54
9.15.4.5	geom	54
9.15.4.6	graphComm	54
9.15.4.7	maxseedlevel	55
9.15.4.8	nbrsOfNbrs	55
9.15.4.9	npx	55
9.15.4.10	npv	55
9.15.4.11	npz	55
9.15.4.12	recvtag	55
9.15.4.13	seeds	55
9.15.4.14	sendtag	55
9.15.4.15	trees	55
9.15.4.16	zoltan_out	55
9.15.4.17	zz	55
9.16	Tree< N, value > Class Template Reference	57
9.16.1	Detailed Description	59
9.16.2	Constructor & Destructor Documentation	59
9.16.2.1	Tree	59
9.16.2.2	Tree	59
9.16.2.3	Tree	59
9.16.2.4	~Tree	59
9.16.3	Member Function Documentation	59
9.16.3.1	addToDerefineList	59
9.16.3.2	addToList	59
9.16.3.3	begin	60
9.16.3.4	centroid	60
9.16.3.5	centroidFixedLevel	60

9.16.3.6	clearMesh	60
9.16.3.7	clearMortonSTL	60
9.16.3.8	clearRefineList	60
9.16.3.9	construct	60
9.16.3.10	constructHigherLevelNbrs	60
9.16.3.11	constructNonlocalHigherLevelNbrs	60
9.16.3.12	convertCoordToMorton	60
9.16.3.13	convertStl2Morton	60
9.16.3.14	count	61
9.16.3.15	Dbegin	61
9.16.3.16	Dend	61
9.16.3.17	derefine	61
9.16.3.18	derefineDerefineList	61
9.16.3.19	derefineDerefineList	61
9.16.3.20	enclosingBox	61
9.16.3.21	enclosingBoxFixedLevel	61
9.16.3.22	end	61
9.16.3.23	extractBoundary	62
9.16.3.24	extractBoundaryP	62
9.16.3.25	find	62
9.16.3.26	findFlipLevel	62
9.16.3.27	findInDerefine	62
9.16.3.28	findInList	62
9.16.3.29	flipForNbr	62
9.16.3.30	flipRefineElemTag	62
9.16.3.31	fourToOne	62
9.16.3.32	fourToOneP	63
9.16.3.33	getDirections	63
9.16.3.34	getKey	63
9.16.3.35	insertKey	63
9.16.3.36	insertNbrs	63
9.16.3.37	insertSeed	63
9.16.3.38	isBoundary	63
9.16.3.39	isBoundary	63
9.16.3.40	isInMeshList	63
9.16.3.41	isInRefineList	63

9.16.3.42	isInsideSolid	63
9.16.3.43	IsInVectorList	64
9.16.3.44	level	64
9.16.3.45	mortonSTLclear	64
9.16.3.46	printMesh	64
9.16.3.47	pushToDerefinelist	64
9.16.3.48	pushToRefinelist	64
9.16.3.49	Rbegin	64
9.16.3.50	readDerefineList	64
9.16.3.51	readRefineList	64
9.16.3.52	refine	64
9.16.3.53	refinelistReset	65
9.16.3.54	refineListSize	65
9.16.3.55	refineRefineList	65
9.16.3.56	refineRefineList	65
9.16.3.57	refineRefineList	65
9.16.3.58	removeFromDerefineList	65
9.16.3.59	Rend	65
9.16.3.60	reserve	65
9.16.3.61	retainFourToOne	65
9.16.3.62	siblings	65
9.16.3.63	size	65
9.16.4	Friends And Related Function Documentation	66
9.16.4.1	Hdf5Xmf	66
9.16.5	Member Data Documentation	66
9.16.5.1	ancestorcoords	66
9.16.5.2	ancestorkey	66
9.16.5.3	ancestorlength	66
9.16.5.4	boundarylist	66
9.16.5.5	derefinelist	66
9.16.5.6	mesh	66
9.16.5.7	mortonSTL	66
9.16.5.8	npx	67
9.16.5.9	npz	67
9.16.5.10	npz	67
9.16.5.11	refinelist	67

9.17	stl::triangle Struct Reference	68
9.17.1	Detailed Description	68
9.17.2	Constructor & Destructor Documentation	68
9.17.2.1	triangle	68
9.17.3	Member Data Documentation	68
9.17.3.1	normal	68
9.17.3.2	v1	68
9.17.3.3	v2	68
9.17.3.4	v3	68
9.18	Vector3 Class Reference	69
9.18.1	Constructor & Destructor Documentation	69
9.18.1.1	Vector3	69
9.18.1.2	Vector3	69
9.18.1.3	~Vector3	69
9.18.2	Member Function Documentation	69
9.18.2.1	Length	69
9.18.2.2	Normalize	69
9.18.2.3	Vectors	69
9.18.3	Member Data Documentation	69
9.18.3.1	X	69
9.18.3.2	Y	69
9.18.3.3	Z	69
9.19	Voxel< N, value > Class Template Reference	70
9.19.1	Detailed Description	70
9.19.2	Constructor & Destructor Documentation	70
9.19.2.1	Voxel	70
9.19.2.2	~Voxel	70
9.19.3	Member Function Documentation	71
9.19.3.1	checkSiblingStatus	71
9.19.3.2	derefineGeomTree	71
9.19.3.3	distributeGeomToLeaves	71
9.19.3.4	generateSearchTree	71
9.19.3.5	IsInsideSegment	71
9.19.3.6	setLevel	71
9.19.4	Friends And Related Function Documentation	71
9.19.4.1	Hdf5XmfV	71

9.19.5	Member Data Documentation	71
9.19.5.1	lookup	71
9.19.5.2	maxlevel	71
9.19.5.3	mesh	72
9.19.5.4	numMax	72
9.20	Zoltan_Out Struct Reference	73
9.20.1	Detailed Description	73
9.20.2	Member Data Documentation	74
9.20.2.1	changes	74
9.20.2.2	exportGlobalGids	74
9.20.2.3	exportLocalGids	74
9.20.2.4	exportProcs	74
9.20.2.5	exportToPart	74
9.20.2.6	importGlobalGids	74
9.20.2.7	importLocalGids	74
9.20.2.8	importProcs	74
9.20.2.9	importToPart	74
9.20.2.10	numExport	74
9.20.2.11	numGidEntries	74
9.20.2.12	numImport	74
9.20.2.13	numLidEntries	74
9.20.2.14	parts	74
10	File Documentation	75
10.1	/home/jhasbestan/Morton_Parallel_v0/src/communicate.cpp File Reference	75
10.1.1	Function Documentation	75
10.1.1.1	ConvertType	75
10.2	/home/jhasbestan/Morton_Parallel_v0/src/forest.cpp File Reference	76
10.2.1	Define Documentation	76
10.2.1.1	METHOD	76
10.2.1.2	REORDER	76
10.2.1.3	SENDBOOL	76
10.3	/home/jhasbestan/Morton_Parallel_v0/src/Ftree_top.cpp File Reference	77
10.3.1	Define Documentation	77
10.3.1.1	PROCSIZE	77
10.3.1.2	TREESIZE	77
10.3.2	Function Documentation	77

10.3.2.1	fTreeProessorTopology	77
10.4	/home/jhasbestan/Morton_Parallel_v0/src/full_tree.cpp File Reference	78
10.5	/home/jhasbestan/Morton_Parallel_v0/src/include/communicate.h File Reference	79
10.6	/home/jhasbestan/Morton_Parallel_v0/src/include/datatype.h File Reference	80
10.6.1	Function Documentation	81
10.6.1.1	getAbstractionDataType	81
10.6.1.2	getAbstractionDataType< char >	81
10.6.1.3	getAbstractionDataType< double >	81
10.6.1.4	getAbstractionDataType< float >	81
10.6.1.5	getAbstractionDataType< int >	81
10.6.1.6	getAbstractionDataType< long >	81
10.6.1.7	getAbstractionDataType< nullptr_t >	81
10.6.1.8	getAbstractionDataType< short >	81
10.6.1.9	getAbstractionDataType< unsigned char >	81
10.6.1.10	getAbstractionDataType< unsigned int >	81
10.6.1.11	getAbstractionDataType< unsigned long >	81
10.6.1.12	getAbstractionDataType< unsigned short >	81
10.7	/home/jhasbestan/Morton_Parallel_v0/src/include/definitions.h File Reference	82
10.7.1	Define Documentation	83
10.7.1.1	BLUE	83
10.7.1.2	CYAN	83
10.7.1.3	GREEN	83
10.7.1.4	hash	83
10.7.1.5	MAGENTA	83
10.7.1.6	nonnative	83
10.7.1.7	RED	83
10.7.1.8	RESET	83
10.7.1.9	WSIZE	83
10.7.1.10	YELLOW	83
10.8	/home/jhasbestan/Morton_Parallel_v0/src/include/forest.h File Reference	84
10.9	/home/jhasbestan/Morton_Parallel_v0/src/include/parse_stl.h File Reference	85
10.9.1	Function Documentation	85
10.9.1.1	checkMesh	85
10.10	/home/jhasbestan/Morton_Parallel_v0/src/include/phdf5.h File Reference	86
10.11	/home/jhasbestan/Morton_Parallel_v0/src/include/scale.h File Reference	87
10.12	/home/jhasbestan/Morton_Parallel_v0/src/include/templateForest.h File Reference	88

10.13/home/jhasbestan/Morton_Parallel_v0/src/include/tree.h File Reference	89
10.14/home/jhasbestan/Morton_Parallel_v0/src/include/typedefs.h File Reference	90
10.14.1 Define Documentation	91
10.14.1.1 DEBUG	91
10.14.1.2 Rma	91
10.14.2 Typedef Documentation	91
10.14.2.1 Center_coords	91
10.14.3 Function Documentation	91
10.14.3.1 fTreeProessorTopology	91
10.14.3.2 readSTLGeom	91
10.14.3.3 treeProcessorTopology	91
10.14.3.4 TwoPowN	91
10.14.3.5 zoltanGeometricPartitioner	91
10.14.3.6 zoltanGeometricPartitionerSerial	91
10.15/home/jhasbestan/Morton_Parallel_v0/src/main.cpp File Reference	92
10.15.1 Define Documentation	92
10.15.1.1 PROCSIZE	92
10.15.1.2 TREESIZE	92
10.15.1.3 WEAK	92
10.15.1.4 WEIGHT	92
10.15.1.5 ZOLTAN_GEOMETRIC_PARTITION	92
10.15.1.6 ZOLTAN_ON	92
10.15.2 Function Documentation	92
10.15.2.1 main	92
10.16/home/jhasbestan/Morton_Parallel_v0/src/parse_stl.cpp File Reference	93
10.16.1 Define Documentation	93
10.16.1.1 CHECK_MESH	93
10.16.1.2 MYSCALE	93
10.16.2 Function Documentation	93
10.16.2.1 checkMesh	93
10.16.2.2 readSTLGeom	93
10.17/home/jhasbestan/Morton_Parallel_v0/src/phdf5.cpp File Reference	94
10.17.1 Define Documentation	94
10.17.1.1 H5FILE	94
10.17.1.2 H5FILE_NAME	94
10.17.1.3 offset0	94

10.17.1.4 offset1	94
10.17.1.5 XDMF_NAME	94
10.17.2 Function Documentation	94
10.17.2.1 integer_string	94
10.18/home/jhasbestan/Morton_Parallel_v0/src/scale.cpp File Reference	95
10.19/home/jhasbestan/Morton_Parallel_v0/src/templateForest.cpp File Reference	96
10.19.1 Define Documentation	96
10.19.1.1 METHOD	96
10.19.1.2 OVERLAP	96
10.19.1.3 REORDER	96
10.19.1.4 SENDBOOL	96
10.20/home/jhasbestan/Morton_Parallel_v0/src/tree.cpp File Reference	97
10.21/home/jhasbestan/Morton_Parallel_v0/src/tree_proc.cpp File Reference	98
10.21.1 Define Documentation	98
10.21.1.1 PROCSIZE	98
10.21.1.2 TREESIZE	98
10.21.1.3 WEIGHT	98
10.21.1.4 ZOLTAN_GEOMETRIC_PARTITION	98
10.21.1.5 ZOLTAN_ON	98
10.21.2 Function Documentation	98
10.21.2.1 treeProcessorTopology	98
10.22/home/jhasbestan/Morton_Parallel_v0/src/voxel.cpp File Reference	99
10.23/home/jhasbestan/Morton_Parallel_v0/src/zoltan.cpp File Reference	100
10.23.1 Define Documentation	101
10.23.1.1 TOL	101
10.23.2 Function Documentation	101
10.23.2.1 get_geometry_list	101
10.23.2.2 get_num_geometry	101
10.23.2.3 get_number_of_objects	101
10.23.2.4 get_object_list	101
10.23.2.5 zoltanGeometricPartitioner	101
10.23.2.6 zoltanGeometricPartitionerSerial	101

Chapter 1

Main Page

"Paralle implementation of Forest of Octrees using Morton Encoding"

Part of NSF project: GEM3D

Required Libraries: HDF5, MPI, Zoltan, CMAKE

If the [.stl](#) file is too large to open with one processor, it is recommended to use a coarser version for generating processor topology, and then each processor can open the big file in parallel and only read the portion related to them

Usage: progName input/geometry *.[stl](#) <params.txt

Authors:

Dr. Jaber J. Hasbestan, Dr. Inanc Senocak
PI: Inanc Senocak

Date:

12 Feb 2017

Copyright (c) 2017 Swanson School of Engineering Department of Mechanical Engineering and Material Sciences Boise State University

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Chapter 2

Directory Hierarchy

2.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

src	14
include	13

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Abstraction	15
stl	16

Chapter 4

Class Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

CenterCoords	17
CommCollective< Type >	18
CommPoint2Point< Type >	21
DataType	24
Forest< N, Nvalue, M, Mvalue >	25
FullOctreeTop< N >	34
GraphData	40
MeshData	41
Message	42
MpiCom	44
Phdf5< N, Nvalue, M, Mvalue >	45
stl::point	47
stl::stl_data	48
TemplateForest< N, Nvalue, M, Mvalue, T >	49
Tree< N, value >	57
FullTree< N, value >	36
Voxel< N, value >	70
stl::triangle	68
Vector3	69
Zoltan_Out	73

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

CenterCoords (Stores the coordinate of the centroid of the elements)	17
CommCollective< Type > (Template wrapper around MPI functions for collective communications)	18
CommPoint2Point< Type > (A template wrapper around MPI functions for point to point communication)	21
DataType (This class abstracts the MPI_Datatypes)	24
Forest< N, Nvalue, M, Mvalue > (Template class that is a forest of octrees with semi-structured process topology)	25
FullOctreeTop< N >	34
FullTree< N, value > (This Class is specifically designed for weak analysis to operate on the full-tree topology in this class, level is preset by the user and the level function is redefined to make use of the polymorphism)	36
GraphData (Struct to supply information required by zoltan for geometric partitioners)	40
MeshData (Struct to supply information required by zoltan for geometric partitioners)	41
Message (Struct that embeds data related to the message and envelope)	42
MpiCom (Class for embedding data related to the communicator)	44
Phdf5< N, Nvalue, M, Mvalue > (This Writes out Tree data in hdf5 format in parallel with *.xmf as metadata suitable for paraview and visit)	45
stl::point (Structure to hold coordinates of a point)	47
stl::stl_data (Structure to store vector of triangles read in from *.stl file)	48
TemplateForest< N, Nvalue, M, Mvalue, T >	49
Tree< N, value > (This Class Generates a 4:1 balancerd AMR mesh)	57
stl::triangle (Structure to store normals and vertices of a triangle)	68
Vector3	69
Voxel< N, value > (This Class Generates an unbalancerd Voxel to improve search by geometry partitioning)	70
Zoltan_Out (This structure is an interface to store the output from Zoltan)	73

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

/home/jhasbestan/Morton_Parallel_v0/src/ communicate.cpp	75
/home/jhasbestan/Morton_Parallel_v0/src/ forest.cpp	76
/home/jhasbestan/Morton_Parallel_v0/src/ Ftree_top.cpp	77
/home/jhasbestan/Morton_Parallel_v0/src/ full_tree.cpp	78
/home/jhasbestan/Morton_Parallel_v0/src/ main.cpp	92
/home/jhasbestan/Morton_Parallel_v0/src/ parse_stl.cpp	93
/home/jhasbestan/Morton_Parallel_v0/src/ phdf5.cpp	94
/home/jhasbestan/Morton_Parallel_v0/src/ scale.cpp	95
/home/jhasbestan/Morton_Parallel_v0/src/ templateForest.cpp	96
/home/jhasbestan/Morton_Parallel_v0/src/ tree.cpp	97
/home/jhasbestan/Morton_Parallel_v0/src/ tree_proc.cpp	98
/home/jhasbestan/Morton_Parallel_v0/src/ voxel.cpp	99
/home/jhasbestan/Morton_Parallel_v0/src/ zoltan.cpp	100
/home/jhasbestan/Morton_Parallel_v0/src/include/ communicate.h	79
/home/jhasbestan/Morton_Parallel_v0/src/include/ datatype.h	80
/home/jhasbestan/Morton_Parallel_v0/src/include/ definitions.h	82
/home/jhasbestan/Morton_Parallel_v0/src/include/ forest.h	84
/home/jhasbestan/Morton_Parallel_v0/src/include/ parse_stl.h	85
/home/jhasbestan/Morton_Parallel_v0/src/include/ phdf5.h	86
/home/jhasbestan/Morton_Parallel_v0/src/include/ scale.h	87
/home/jhasbestan/Morton_Parallel_v0/src/include/ templateForest.h	88
/home/jhasbestan/Morton_Parallel_v0/src/include/ tree.h	89
/home/jhasbestan/Morton_Parallel_v0/src/include/ typedefs.h	90

Chapter 7

Directory Documentation

7.1 /home/jhasbestan/Morton_Parallel_v0/src/include/ Directory Reference

Files

- file [communicate.h](#)
- file [datatype.h](#)
- file [definitions.h](#)
- file [forest.h](#)
- file [parse_stl.h](#)
- file [phdf5.h](#)
- file [scale.h](#)
- file [templateForest.h](#)
- file [tree.h](#)
- file [typedefs.h](#)

7.2 /home/jhasbestan/Morton_Parallel_v0/src/ Directory Reference

Directories

- directory [include](#)

Files

- file [communicate.cpp](#)
- file [forest.cpp](#)
- file [Ftree_top.cpp](#)
- file [full_tree.cpp](#)
- file [main.cpp](#)
- file [parse_stl.cpp](#)
- file [phdf5.cpp](#)
- file [scale.cpp](#)
- file [templateForest.cpp](#)
- file [tree.cpp](#)
- file [tree_proc.cpp](#)
- file [voxel.cpp](#)
- file [zoltan.cpp](#)

Chapter 8

Namespace Documentation

8.1 Abstraction Namespace Reference

Enumerations

- enum `DataType` {
 `type_byte`, `type_char`, `type_unsigned_char`, `type_short`,
 `type_unsigned_short`, `type_int`, `type_unsigned_int`, `type_long`,
 `type_unsigned_long`, `type_float`, `type_double` }

8.1.1 Enumeration Type Documentation

8.1.1.1 enum `Abstraction::DataType`

Enumerator:

type_byte
type_char
type_unsigned_char
type_short
type_unsigned_short
type_int
type_unsigned_int
type_long
type_unsigned_long
type_float
type_double

8.2 stl Namespace Reference

Classes

- struct [point](#)
Structure to hold coordinates of a [point](#).
- struct [triangle](#)
structure to store normals and vertices of a [triangle](#)
- struct [stl_data](#)
*structure to store vector of triangles read in from *.[stl](#) file*

Functions

- `std::ostream & operator<< (std::ostream &out, const triangle &t)`
- `stl_data parse_stl (const std::string &stl_path)`
- `std::ostream & operator<< (std::ostream &out, const point p)`
- `float parse_float (std::ifstream &s)`
- `point parse_point (std::ifstream &s)`

8.2.1 Function Documentation

8.2.1.1 `std::ostream& stl::operator<< (std::ostream & out, const point p)`

8.2.1.2 `std::ostream & stl::operator<< (std::ostream & out, const triangle & t)`

8.2.1.3 `float stl::parse_float (std::ifstream & s)`

8.2.1.4 `point stl::parse_point (std::ifstream & s)`

8.2.1.5 `stl_data stl::parse_stl (const std::string & stl_path)`

Chapter 9

Class Documentation

9.1 CenterCoords Struct Reference

Stores the coordinate of the centroid of the elements.

```
#include <typedefs.h>
```

Public Attributes

- real [x](#)
- real [y](#)
- real [z](#)

9.1.1 Detailed Description

Stores the coordinate of the centroid of the elements.

9.1.2 Member Data Documentation

9.1.2.1 `real CenterCoords::x`

9.1.2.2 `real CenterCoords::y`

9.1.2.3 `real CenterCoords::z`

The documentation for this struct was generated from the following file:

- `/home/jhasbestan/Morton_Parallel_v0/src/include/typedefs.h`

9.2 CommCollective< Type > Class Template Reference

is a template wrapper around MPI functions for collective communications

```
#include <communicate.h>
```

Public Member Functions

- [CommCollective](#) (void *buf, uint size, uint root)
- [CommCollective](#) (void *buff, uint size)
- void [bcast](#) ()
- void [lbcast](#) ()
- void [getTotalNumber](#) (uint *offset, uint *myvalue, uint *totalvalue)
- void [lgetTotalNumber](#) (uint *offset, uint *myvalue, uint *totalvalue)
- void [waitOnRequest](#) ()
- [~CommCollective](#) ()
- template<>
void [getTotalNumber](#) (uint *offset, uint *myvalue, uint *totalvalue)
- template<>
void [lgetTotalNumber](#) (uint *offset, uint *myvalue, uint *totalvalue)

Private Attributes

- [MpiCom Com](#)
- [Message Msg](#)

9.2.1 Detailed Description

```
template<class Type> class CommCollective< Type >
```

is a template wrapper around MPI functions for collective communications This class is specialized for collective communications and functions

9.2.2 Constructor & Destructor Documentation

9.2.2.1 `template<class Type > CommCollective< Type >::CommCollective (void * buf, uint size, uint root) [inline]`

constructor

Parameters:

root another constructor for Communicator class

9.2.2.2 `template<class Type > CommCollective< Type >::CommCollective (void * buff, uint size) [inline]`

constructor specialized for root=comsize-1

Parameters:

size another constructor for Communicator class, this one sets the root as comsize-1 for hdf5

9.2.2.3 `template<class Type > CommCollective< Type >::~~CommCollective () [inline]`

9.2.3 Member Function Documentation

9.2.3.1 `template<class Type > void CommCollective< Type >::bcast () [inline]`

blockin broadcast

9.2.3.2 `template<> void CommCollective< uint >::getTotalNumber (uint * offset, uint * myvalue, uint * totalvalue) [inline]`

< the largest offset belongs to the processor with highest rank, add this to its number of cubes will give us the total value

9.2.3.3 `template<class Type > void CommCollective< Type >::getTotalNumber (uint * offset, uint * myvalue, uint * totalvalue)`

calculates the world population for a given variable

9.2.3.4 `template<class Type > void CommCollective< Type >::Ibcast () [inline]`

non-blocking broadcast

I speozialized this function since this is only needed for unsigned ints

9.2.3.5 `template<> void CommCollective< uint >::IgetTotalNumber (uint * offset, uint * myvalue, uint * totalvalue) [inline]`

< the largest offset belongs to the processor with highest rank, add this to its number of cubes will give us the total value

9.2.3.6 `template<class Type > void CommCollective< Type >::IgetTotalNumber (uint * offset, uint * myvalue, uint * totalvalue)`

calculates the world population for a given variable in a non-blocking fashion

9.2.3.7 `template<class Type > void CommCollective< Type >::waitOnRequest () [inline]`

9.2.4 Member Data Documentation

9.2.4.1 `template<class Type > MPICom CommCollective< Type >::Com [private]`

9.2.4.2 `template<class Type > Message CommCollective< Type >::Msg [private]`

The documentation for this class was generated from the following files:

- [/home/jhasbestan/Morton_Parallel_v0/src/include/communicate.h](#)
- [/home/jhasbestan/Morton_Parallel_v0/src/communicate.cpp](#)

9.3 CommPoint2Point< Type > Class Template Reference

A template wrapper around MPI functions for point to point communication.

```
#include <communicate.h>
```

Public Member Functions

- [CommPoint2Point](#) (void *buff, uint size, uint *tg, uint snd, uint rcv, uint type, MPI_Comm com)
- [CommPoint2Point](#) (void *buff, uint size, uint snd, uint rcv)
- [CommPoint2Point](#) (void *buff, uint size)
- void [assignSender](#) (uint sndr)
- void [assignReceiver](#) (uint rcv)
- integer [myRank](#) ()
- integer [mySize](#) ()
- void [Irecv](#) ()
- void [Isend](#) ()
- void [recv](#) ()
- void [send](#) ()
- void [getOffset](#) (uint myvalue, uint *offset)
- [~CommPoint2Point](#) ()

Private Attributes

- [MpiCom Com](#)
- [Message Msg](#)

9.3.1 Detailed Description

`template<class Type> class CommPoint2Point< Type >`

A template wrapper around MPI functions for point to point communication. class [CommPoint2Point](#) abstracts point to point communications and functions

9.3.2 Constructor & Destructor Documentation

9.3.2.1 `template<class Type > CommPoint2Point< Type >::CommPoint2Point (void * buff, uint size, uint * tg, uint snd, uint rcv, uint type, MPI_Comm Comm) [inline]`

full constructor

< default tag for message send and receive is "0", MPI_ANY_TAG is only OK for receive command so it can not be used here

Parameters:

Comm If communicator is not specified it will be set as MPI_COMM_WORLD by default

9.3.2.2 `template<class Type > CommPoint2Point< Type >::CommPoint2Point (void * buff, uint size, uint snd, uint rcv) [inline]`

default constructor

Parameters:

rcv another constructor for Communicator class

9.3.2.3 `template<class Type > CommPoint2Point< Type >::CommPoint2Point (void * buff, uint size) [inline]`

deferring the assignment of sender and receiver

Parameters:

size another constructor for Communicator class, defers assigning sender and receiver

9.3.2.4 `template<class Type > CommPoint2Point< Type >::~~CommPoint2Point () [inline]`

9.3.3 Member Function Documentation

9.3.3.1 `template<class Type > void CommPoint2Point< Type >::assignReceiver (uint rcv) [inline]`

assigns the sender of the message

9.3.3.2 `template<class Type > void CommPoint2Point< Type >::assignSender (uint sndr) [inline]`

9.3.3.3 `template<class Type > void CommPoint2Point< Type >::getOffset (uint myvalue, uint * offset) [inline]`

calculates the offset of each processor for variable myvalue

Parameters:

offset gets the number of elements before the current processor, it is needed for globally defining the elements

9.3.3.4 `template<class Type > void CommPoint2Point< Type >::Irecv () [inline]`

non-blocking receive

9.3.3.5 `template<class Type > void CommPoint2Point< Type >::Isend () [inline]`

non-blocking send

9.3.3.6 template<class Type > int CommPoint2Point< Type >::myRank () [inline]

assigns the destination the message gets the rank of the processor

9.3.3.7 template<class Type > int CommPoint2Point< Type >::mySize () [inline]**9.3.3.8 template<class Type > void CommPoint2Point< Type >::recv () [inline]**

blocking receive

9.3.3.9 template<class Type > void CommPoint2Point< Type >::send () [inline]

blocking send

9.3.4 Member Data Documentation**9.3.4.1 template<class Type > MPI_Comm CommPoint2Point< Type >::Com [private]****9.3.4.2 template<class Type > Message CommPoint2Point< Type >::Msg [private]**

The documentation for this class was generated from the following files:

- /home/jhasbestan/Morton_Parallel_v0/src/include/[communicate.h](#)
- /home/jhasbestan/Morton_Parallel_v0/src/[communicate.cpp](#)

9.4 DataType Class Reference

this class abstracts the MPI_Datatypes

```
#include <datatype.h>
```

9.4.1 Detailed Description

this class abstracts the MPI_Datatypes using template initializations one can template MPI functions using template specialization for further details see, <https://chuckaknight.wordpress.com/2013/03/13/intrinsic-type-conversion-using-templates/> since BYTE is not a native type in C++, for MPI_TYPE, the nullptr_t (null pointer type) is utilized

The documentation for this class was generated from the following file:

- /home/jhasbestan/Morton_Parallel_v0/src/include/[datatype.h](#)

9.5 Forest< N, Nvalue, M, Mvalue > Class Template Reference

template class that is a forest of octrees with semi-structured process topology

```
#include <forest.h>
```

Public Member Functions

- [Forest](#) (real *length, real *coords, [Tree](#)< M, Mvalue > &proc, const int fixedlevel, uint nx, uint ny, uint nz)
- void [assignSeeds](#) (real *length, [Tree](#)< M, Mvalue > &proc, const int fixedlevel)
- uint [getTotalSize](#) ()
- void [refineEachTreeVoxel](#) (uint nlevel)
- void [assignGeom](#) ([Tree](#)< M, Mvalue > &proc, const uint fixedlevel, real *geom_xyz, uint geom_nn)
- void [refineEachTree](#) (uint nlevel)
- void [getListEachTree](#) ()
- void [fourToOneBalance](#) ([Tree](#)< M, uint > &proc)
- bool [isInSeed](#) (morton< M > &key, uint *counter)
- void [flipAll](#) (morton< N > &key, uint *mylevel, uint *direction)
- void [findFlipLevel](#) (morton< N+M > &key, uint *mylevel, uint *changedirectionlevel, uint *direction)
- void [flipForNbr](#) (morton< N+M > &key, uint *mylevel, uint *changedirectionlevel, uint *direction)
- void [getNbrSeedLevel](#) (morton< N+M > &combinedkey, uint topologylevel, uint *nbrseedlevel, [Tree](#)< M, uint > &proc)
- void [debug](#) ([Tree](#)< M, Mvalue > &proc)
- void [getElemNbrs](#) ([Tree](#)< M, Mvalue > &proc, const morton< M > key, bitvector< M > &nbr)
- void [comPatternConstruct](#) ([Tree](#)< M, Mvalue > &proc)
- void [getDirections](#) (morton< N+M > &key, uint combinedlevel, vector< uint > &directions)
- void [encodeGeometry](#) ()
- void [removeAllZeroSingularity](#) (morton< N+M > &key, const uint &combinedlevel)
- void [getMaxSeedsLevel](#) ([Tree](#)< M, Mvalue > &proc)
- void [findSeedLevelForRcvdMessage](#) (const morton< N+M > &key, uint *mylevel, [Tree](#)< M, Mvalue > &proc)
- void [recoverAllZeroSingularity](#) (morton< N+M > &key, const uint &combinedlevel)
- void [constructSeedKeyForRcvdMessage](#) (const morton< N+M > &key, const uint &seedlevel, morton< M > &seedkey)
- void [constructElementKeyForRcvdMessage](#) (const morton< N+M > &key, const uint &seedlevel, morton< N > &elementkey)
- void [refineForestBalanced](#) (uint nlevel, [Tree](#)< M, Mvalue > &proc)
- void [combinedLevel](#) (const morton< N+M > &key, uint *level)
- void [zoltanGeomrepart](#) ([Tree](#)< M, Mvalue > &proc, uint setmethod)
- uint [forestsize](#) ()
- void [retainFourToOneBalance](#) ([Tree](#)< M, uint > &proc)
- void [moveGeom](#) ([Tree](#)< M, Mvalue > &proc, const uint fixedlevel, real *geom_xyz, uint n, real x[3])
- void [pushToDerefineEachTree](#) (uint nlevel, [Tree](#)< M, uint > &proc)
- void [convertBitsToDouble](#) (morton< N+M > &key, double *val)
- void [convertDoubleToBits](#) (morton< N+M > &key, const double val)
- void [createCommGraph](#) (uint Nnbr)
- void [createNbrsOfNbrs](#) ()
- void [debugDerefine](#) ([Tree](#)< M, Mvalue > &proc)

- void [checkGraphConsistency](#) ()
- void [checkNbrsOfNbrsConsistency](#) ()
- bool [checkWithNbrs](#) (bool *sendbuf, bool *recvbuf)
- void [rcvrMessageSize](#) (int *sendbuf, int *recvbuf)
- void [getTotalMeshSize](#) ()
- void [checkZoltanPartConsistency](#) (Tree< M, Mvalue > &proc)
- void [constructCommWeak](#) (const vector< int >Nbr)
- [~Forest](#) ()

Protected Attributes

- real [ancestorlength](#) [3]
- real [ancestorcoords](#) [3]
- uint [npz](#)
- uint [npz](#)
- uint [npz](#)

Private Attributes

- [MpiCom Com](#)
- treelist< N, Nvalue > [trees](#)
- bitlist< M > [seeds](#)
- uint [maxseedlevel](#)
- Tree< M, real > [geom](#)
- vector< uint > [destination](#)
- vector< uint > [nbrsOfNbrs](#)
- struct Zoltan_Struct * [zz](#) = nullptr
- [Zoltan_Out zoltan_out](#)
- MPI_Comm [graphComm](#)

Friends

- class [Phdf5](#)

9.5.1 Detailed Description

template<size_t N, typename Nvalue, size_t M, typename Mvalue> class Forest< N, Nvalue, M, Mvalue >

template class that is a forest of octrees with semi-structured process topology includes a list of tree's and functionality for manipulation as well as exchange of the trees with neighboring processes The algorithm starts with a very coarse 16 bit semi-structured processor topology, for now only 16 bits are used but it can be modified according to the need each process will have one forest and trees will be distributed dynamically as the solution progresses

9.5.2 Constructor & Destructor Documentation

9.5.2.1 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> Forest< N, Nvalue, M, Mvalue >::Forest (real * length, real * coords, Tree< M, Mvalue > & proc, const int fixedlevel, uint nx, uint ny, uint nz) [inline]`

constructor

<part I, initialize the ancestor coords and length, just like we did for class tree

9.5.2.2 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> Forest< N, Nvalue, M, Mvalue >::~~Forest () [inline]`

9.5.3 Member Function Documentation

9.5.3.1 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue >::assignGeom (Tree< M, Mvalue > & proc, const uint fixedlevel, real * geom_xyz, uint geom_nn) [inline]`

9.5.3.2 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue >::assignSeeds (real * length, Tree< M, Mvalue > & proc, const int fixedlevel) [inline]`

9.5.3.3 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue >::checkGraphConsistency () [inline]`

9.5.3.4 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue >::checkNbrsOfNbrsConsistency () [inline]`

9.5.3.5 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> bool Forest< N, Nvalue, M, Mvalue >::checkWithNbrs (bool * sendbuf, bool * recvbuf) [inline]`

this is to enforce broadcast only with first degree neighbors

Parameters:

recvbuf creates distributed (acalable) graph for communication

9.5.3.6 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue >::checkZoltanPartConsistency (Tree< M, Mvalue > & proc) [inline]`

Parameters:

proc creates distributed (acalable) graph for communication

9.5.3.7 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue >::combinedLevel (const morton< N+M > & key, uint * level) [inline]`

to prevent unnecessary bit operation, the morton code is placed from starting from left hand side

9.5.3.8 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::comPatternConstruct (Tree< M, Mvalue > & proc) [inline]`

9.5.3.9 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::constructCommWeak (const vector< int > Nbr) [inline]`

Parameters:

Nbr update communication based on Nbr provided by class fullTreeTopology

9.5.3.10 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::constructElementKeyForRcvdMessage (const morton< N+M > & key, const uint & seedlevel, morton< N > & elementkey) [inline]`

9.5.3.11 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::constructSeedKeyForRcvdMessage (const morton< N+M > & key, const uint & seedlevel, morton< M > & seedkey) [inline]`

9.5.3.12 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue >::convertBitsToDouble (morton< N+M > key, double * val)`

9.5.3.13 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue >::convertDoubleToBits (morton< N+M > & key, const double val)`

9.5.3.14 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::createCommGraph (uint Lnbr) [inline]`

level of neighbors

Parameters:

Lnbr creates distributed (scalable) graph for communication

9.5.3.15 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::createNbrsOfNbrs () [inline]`

9.5.3.16 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::debug (Tree< M, Mvalue > & proc) [inline]`

9.5.3.17 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::debugDerefine (Tree< M, Mvalue > & proc) [inline]`

9.5.3.18 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::encodeGeometry () [inline]`

9.5.3.19 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::findFlipLevel (morton< N+M > key, uint * mylevel, uint * changedirectionlevel, uint * direction) [inline]`

same as the function defined in class three except that it worked on (M+N) bits

9.5.3.20 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::findSeedLevelForRcvdMessage (const morton< N+M > & key, uint * mylevel, Tree< M, Mvalue > & proc) [inline]`

9.5.3.21 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::flipAll (morton< N > & key, uint * mylevel, uint * direction) [inline]`

9.5.3.22 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::flipForNbr (morton< N+M > & key, uint * mylevel, uint * changedirectionlevel, uint * direction) [inline]`

same as the function defined in class three except that it workd on (M+N) bits

9.5.3.23 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > uint Forest< N, Nvalue, M, Mvalue >::forestsize () [inline]`

9.5.3.24 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::fourToOneBalance (Tree< M, uint > & proc) [inline]`

4:1 balance enforced at forest including the other processors

9.5.3.25 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::getDirections (morton< N+M > & key, uint combinedlevel, vector< uint > & directions) [inline]`

9.5.3.26 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::getElemNbrs (Tree< M, Mvalue > & proc, const morton< M > key, bitvector< M > & nbr) [inline]`

collects the neighbors of a given element

9.5.3.27 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::getListEachTree () [inline]`

9.5.3.28 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::getMaxSeedsLevel (Tree< M, Mvalue> & proc) [inline]`

9.5.3.29 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::getNbrSeedLevel (morton< N+M> & combinedkey, uint topologylevel, uint * nbrseedleve, Tree< M, uint> & proc) [inline]`

9.5.3.30 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::getTotalMeshSize () [inline]`

9.5.3.31 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> uint Forest< N, Nvalue, M, Mvalue>::getTotalSize () [inline]`

9.5.3.32 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> bool Forest< N, Nvalue, M, Mvalue>::isInSeed (morton< M> & key, uint * counter) [inline]`

check and see if the forest incldues the particluar seed

9.5.3.33 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::moveGeom (Tree< M, Mvalue> & proc, const uint fixedlevel, real * geom_xyz, uint n, real x[3]) [inline]`

moves the geomerty with displacements specified in *x*[3] in x,y and z directions

9.5.3.34 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::pushToDerefineEachTree (uint nlevel, Tree< M, uint> & proc) [inline]`

9.5.3.35 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::rcvrMessageSize (int * sendbuf, int * recvbuf) [inline]`

Parameters:

recvbuf creates distributed (acalable) graph for communication

9.5.3.36 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::recoverAllZeroSingularity (morton< N+M> & key, const uint & combinedlevel) [inline]`

note that this function operates on the element key, this is done to remove redundant calc

9.5.3.37 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::refineEachTree (uint nlevel) [inline]`

Refines every [Tree](#) in the list, *nlevels*, balance is satisfired for each tree but not in the global scope

9.5.3.38 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::refineEachTreeVoxel (uint nlevel) [inline]`

Refines every [Tree](#) in the list, nlevels, balance is satisfied for each tree but not in the boundaries with other trees

9.5.3.39 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::refineForestBalanced (uint nlevel, Tree< M, Mvalue > & proc) [inline]`

9.5.3.40 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::removeAllZeroSingularity (morton< N+M > & key, const uint & combinedlevel) [inline]`

9.5.3.41 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::retainFourToOneBalance (Tree< M, uint > & proc) [inline]`

9.5.3.42 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::zoltanGeomrepart (Tree< M, Mvalue > & proc, uint setmethod) [inline]`

9.5.4 Friends And Related Function Documentation

9.5.4.1 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> friend class Phdf5 [friend]`

9.5.5 Member Data Documentation

9.5.5.1 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> real Forest< N, Nvalue, M, Mvalue >::ancestorcoords[3] [protected]`

centroid of the of the first generation (root) element

9.5.5.2 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> real Forest< N, Nvalue, M, Mvalue >::ancestorlength[3] [protected]`

original length of the first generation (root) element

9.5.5.3 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> MPICom Forest< N, Nvalue, M, Mvalue >::Com` `[private]`

9.5.5.4 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> vector<uint> Forest< N, Nvalue, M, Mvalue >::destination` `[private]`

9.5.5.5 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> Tree<M, real> Forest< N, Nvalue, M, Mvalue >::geom` `[private]`

9.5.5.6 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> MPI_Comm Forest< N, Nvalue, M, Mvalue >::graphComm` `[private]`

Each seed will contain its own geometry points to search, this assumption implicitly coincides processor topology with geometry voxelization this way tree's root will be morton code used in geometry voxelization and no extra operation is necessary

9.5.5.7 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> uint Forest< N, Nvalue, M, Mvalue >::maxseedlevel` `[private]`

finds the maximum level of the seeds

9.5.5.8 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> vector<uint> Forest< N, Nvalue, M, Mvalue >::nbrsOfNbrs` `[private]`

9.5.5.9 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> uint Forest< N, Nvalue, M, Mvalue >::npx` `[protected]`

discretization in x direction, this value for proc tree is 2, therefore forest needs its own value of npx

9.5.5.10 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> uint Forest< N, Nvalue, M, Mvalue >::npy` `[protected]`

discretization in y direction

9.5.5.11 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> uint Forest< N, Nvalue, M, Mvalue >::npz` `[protected]`

discretization in z direction

9.5.5.12 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> bitlist<M> Forest< N, Nvalue, M, Mvalue >::seeds` `[private]`

seeds: morton code for boxes to grow tree

9.5.5.13 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> treelist<N, Nvalue> Forest< N, Nvalue, M, Mvalue >::trees` `[private]`

list of trees that each processor includes

9.5.5.14 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> Zoltan_Out Forest< N, Nvalue, M, Mvalue >::zoltan_out [private]`

9.5.5.15 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> struct Zoltan_Struct* Forest< N, Nvalue, M, Mvalue >::zz = nullptr [read, private]`

The documentation for this class was generated from the following files:

- [/home/jhasbestan/Morton_Parallel_v0/src/include/forest.h](#)
- [/home/jhasbestan/Morton_Parallel_v0/src/forest.cpp](#)

9.6 FullOctreeTop< N > Class Template Reference

```
#include <scale.h>
```

Public Member Functions

- [FullOctreeTop](#) ()
- void [convertRank2Bits](#) (int myrank)
- void [constructNbrProcs](#) ()
- bool [isBoundary](#) (uint &direction)
- void [checkGraphConsistency](#) (int myrank)
- void [readRoot](#) (morton< N > &key)
- void [readNbrs](#) (vector< int > &Nbrs)
- [~FullOctreeTop](#) ()

Private Attributes

- uint [fullOctreeLevel](#)
- morton< N > [rootKey](#)
- vector< int > [Nbr](#)

```
template<size_t N> class FullOctreeTop< N >
```

9.6.1 Constructor & Destructor Documentation

9.6.1.1 `template<size_t N> FullOctreeTop< N >::FullOctreeTop () [inline]`

9.6.1.2 `template<size_t N> FullOctreeTop< N >::~~FullOctreeTop () [inline]`

9.6.2 Member Function Documentation

9.6.2.1 `template<size_t N> void FullOctreeTop< N >::checkGraphConsistency (int myrank) [inline]`

Parameters:

myrank check symmetry of the communication graph, if not symmetric, due to the use of MPI_Probe it will deadlock

9.6.2.2 `template<size_t N> void FullOctreeTop< N >::constructNbrProcs () [inline]`

9.6.2.3 `template<size_t N> void FullOctreeTop< N >::convertRank2Bits (int myrank) [inline]`

9.6.2.4 `template<size_t N> bool FullOctreeTop< N >::isBoundary (uint & direction) [inline]`

9.6.2.5 `template<size_t N> void FullOctreeTop< N >::readNbrs (vector< int > & Nbrs) [inline]`

9.6.2.6 `template<size_t N> void FullOctreeTop< N >::readRoot (morton< N > & key) [inline]`

9.6.3 Member Data Documentation

9.6.3.1 `template<size_t N> uint FullOctreeTop< N >::fullOctreeLevel [private]`

9.6.3.2 `template<size_t N> vector<int> FullOctreeTop< N >::Nbr [private]`

9.6.3.3 `template<size_t N> morton<N> FullOctreeTop< N >::rootKey [private]`

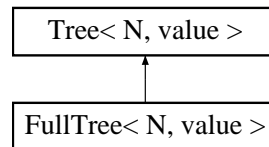
The documentation for this class was generated from the following files:

- `/home/jhasbestan/Morton_Parallel_v0/src/include/scale.h`
- `/home/jhasbestan/Morton_Parallel_v0/src/scale.cpp`

9.7 FullTree< N, value > Class Template Reference

This Class is specifically designed for weak analysis to operate on the fulltree topology in this class, level is preset by the user and the level function is redefined to make use of the polymorphism.

#include <tree.h> Inheritance diagram for FullTree< N, value >::



Public Member Functions

- [FullTree](#) (real *length, real *coords)
- void [setLevel](#) (const uint &l)
- void [level](#) (morton< N > key, uint *level)
- void [insertKey](#) (morton< N > key)
- uint [getLevel](#) ()
- void [nbrsConstrcut](#) (vector< uint > &Nbrs, uint myrank)
- bool [isBoundary](#) (uint &direction, uint myrank)
- bitmap< N, value >::iterator [find](#) (morton< N > key)
- uint [size](#) ()
- void [convertCoordToMorton](#) (real *xyz, morton< N > &key)
- void [assignProcs](#) (vector< uint > &Nbrs, uint myrank)
- bitmap< N, value >::iterator [begin](#) ()
- bitmap< N, value >::iterator [end](#) ()
- [~FullTree](#) ()

Private Attributes

- uint [fixedlevel](#)
- bitmap< N, value > [mesh](#)
- std::unordered_map< morton< N >, int > [refinelist](#)
- bitvector< N > [mortonSTL](#)
- std::unordered_map< morton< N >, int > [derefinelist](#)

9.7.1 Detailed Description

template<size_t N, typename value> class FullTree< N, value >

This Class is specifically designed for weak analysis to operate on the fulltree topology in this class, level is preset by the user and the level function is redefined to make use of the polymorphism.

9.7.2 Constructor & Destructor Documentation

9.7.2.1 `template<size_t N, typename value> FullTree< N, value >::FullTree (real * length, real * coords) [inline]`

9.7.2.2 `template<size_t N, typename value> FullTree< N, value >::~~FullTree () [inline]`

9.7.3 Member Function Documentation

9.7.3.1 `template<size_t N, typename value > void FullTree< N, value >::assignProcs (vector< uint > & Nbrs, uint myrank) [inline]`

9.7.3.2 `template<size_t N, typename value > bitmap< N, value >::iterator FullTree< N, value >::begin () [inline, virtual]`

iterator returning the first object

Reimplemented from [Tree< N, value >](#).

9.7.3.3 `template<size_t N, typename value > void FullTree< N, value >::convertCoordToMorton (real * xyz, morton< N > & key) [inline, virtual]`

!< this function is to find a value given the key

Reimplemented from [Tree< N, value >](#).

9.7.3.4 `template<size_t N, typename value > bitmap< N, value >::iterator FullTree< N, value >::end () [inline, virtual]`

iterator returning the last object

Reimplemented from [Tree< N, value >](#).

9.7.3.5 `template<size_t N, typename value > bitmap< N, value >::iterator FullTree< N, value >::find (morton< N > key) [inline, virtual]`

this function is to find a value given the key

!< this function is to find a value given the key

Reimplemented from [Tree< N, value >](#).

9.7.3.6 `template<size_t N, typename value > uint FullTree< N, value >::getLevel () [inline]`

9.7.3.7 `template<size_t N, typename value > void FullTree< N, value >::insertKey (morton< N > key) [inline]`

Reimplemented from [Tree< N, value >](#).

9.7.3.8 `template<size_t N, typename value> bool FullTree< N, value >::isBoundary (uint & direction, uint myrank) [inline]`

9.7.3.9 `template<size_t N, typename value> void FullTree< N, value >::level (morton< N > key, uint * level) [inline, virtual]`

obtains the level of the element from morton key

to prevent unnecessary bit operation, the morton code is placed from starting from left hand side

now look and see if any siblings exist

Reimplemented from [Tree< N, value >](#).

9.7.3.10 `template<size_t N, typename value> void FullTree< N, value >::nbrsConstrcut (vector< uint > & Nbrs, uint myrank) [inline]`

9.7.3.11 `template<size_t N, typename value> void FullTree< N, value >::setLevel (const uint & l) [inline]`

constructor sets the maximum level for refinement

9.7.3.12 `template<size_t N, typename value> uint FullTree< N, value >::size () [inline, virtual]`

returns the size of the mesh

Reimplemented from [Tree< N, value >](#).

9.7.4 Member Data Documentation

9.7.4.1 `template<size_t N, typename value> std::unordered_map<morton<N>, int> FullTree< N, value >::derefineList [private]`

list of elements tagged to be removed, due to 4:1 balance this list

Reimplemented from [Tree< N, value >](#).

9.7.4.2 `template<size_t N, typename value> uint FullTree< N, value >::fixedlevel [private]`

maximum level of full tree

9.7.4.3 `template<size_t N, typename value> bitmap<N, value> FullTree< N, value >::mesh [private]`

base main container for hashmap

Reimplemented from [Tree< N, value >](#).

Reimplemented from `Tree< N, value >`.

The documentation for this class was generated from the following files:

- /home/jhasbestan/Morton_Parallel_v0/src/include/tree.h
- /home/jhasbestan/Morton_Parallel_v0/src/full_tree.cpp

9.8 GraphData Struct Reference

struct to supply information required by zoltan for geometric partitioners

Public Attributes

- ZOLTAN_ID_TYPE [numMyVertices](#)
- ZOLTAN_ID_PTR [vertexGID](#)
- ZOLTAN_ID_PTR [nbrIndex](#)
- ZOLTAN_ID_PTR [nbrGID](#)
- int * [nbrProc](#)
- float * [c](#)

9.8.1 Detailed Description

struct to supply information required by zoltan for geometric partitioners

9.8.2 Member Data Documentation

9.8.2.1 float* GraphData::c

9.8.2.2 ZOLTAN_ID_PTR GraphData::nbrGID

9.8.2.3 ZOLTAN_ID_PTR GraphData::nbrIndex

9.8.2.4 int* GraphData::nbrProc

9.8.2.5 ZOLTAN_ID_TYPE GraphData::numMyVertices

9.8.2.6 ZOLTAN_ID_PTR GraphData::vertexGID

The documentation for this struct was generated from the following file:

- [/home/jhasbestan/Morton_Parallel_v0/src/zoltan.cpp](#)

9.9 MeshData Struct Reference

struct to supply information required by zoltan for geometric partitioners

Public Attributes

- ZOLTAN_ID_TYPE [numGlobalPoints](#)
- ZOLTAN_ID_TYPE [numMyPoints](#)
- ZOLTAN_ID_PTR [myGlobalIDs](#)
- real * [c](#)
- real * [w](#)

9.9.1 Detailed Description

struct to supply information required by zoltan for geometric partitioners

9.9.2 Member Data Documentation

9.9.2.1 real* MeshData::c

9.9.2.2 ZOLTAN_ID_PTR MeshData::myGlobalIDs

9.9.2.3 ZOLTAN_ID_TYPE MeshData::numGlobalPoints

9.9.2.4 ZOLTAN_ID_TYPE MeshData::numMyPoints

9.9.2.5 real* MeshData::w

The documentation for this struct was generated from the following file:

- [/home/jhasbestan/Morton_Parallel_v0/src/zoltan.cpp](#)

9.10 Message Struct Reference

struct that embeds data related to the message and envelope

```
#include <communicate.h>
```

Public Member Functions

- void [print](#) ()

Public Attributes

- uint [count](#)
- int [tag](#)
- uint [sender](#)
- uint [reciever](#)
- void * [buf](#)
- MPI_Datatype [datatype](#)
- MPI_Status [status](#)
- MPI_Request [request](#)

9.10.1 Detailed Description

struct that embeds data related to the message and envelope

9.10.2 Member Function Documentation

9.10.2.1 void Message::print () [inline]

9.10.3 Member Data Documentation

9.10.3.1 void* Message::buf

pointer to the buffer of the data

9.10.3.2 uint Message::count

count of the message

9.10.3.3 MPI_Datatype Message::datatype

9.10.3.4 uint Message::reciever

who is the destination

9.10.3.5 MPI_Request Message::request**9.10.3.6 uint Message::sender**

who is sending

9.10.3.7 MPI_Status Message::status**9.10.3.8 int Message::tag**

tag of the message

The documentation for this struct was generated from the following file:

- /home/jhasbestan/Morton_Parallel_v0/src/include/[communicate.h](#)

9.11 MpiCom Struct Reference

class for embedding data related to the communicator

```
#include <communicate.h>
```

Public Member Functions

- [MpiCom \(\)](#)
- [MpiCom \(MPI_Comm Com\)](#)

Public Attributes

- MPI_Comm [mpicom](#)
- integer [myrank](#)
- integer [comsize](#)

9.11.1 Detailed Description

class for embedding data related to the communicator

9.11.2 Constructor & Destructor Documentation

9.11.2.1 [MpiCom::MpiCom \(\)](#) [[inline](#)]

9.11.2.2 [MpiCom::MpiCom \(MPI_Comm Com\)](#)

constructor second constructor

9.11.3 Member Data Documentation

9.11.3.1 integer [MpiCom::comsize](#)

size of the communicator

9.11.3.2 MPI_Comm [MpiCom::mpicom](#)

Communicator

9.11.3.3 integer [MpiCom::myrank](#)

rank of the processor

The documentation for this struct was generated from the following file:

- [/home/jhasbestan/Morton_Parallel_v0/src/include/communicate.h](#)

9.12 Phdf5< N, Nvalue, M, Mvalue > Class Template Reference

This Writes out [Tree](#) data in hdf5 format in parallel with *.xmf as metadata suitable for paraview and visit.

```
#include <phdf5.h>
```

Public Member Functions

- [Phdf5](#) ()
- void [writePolyvertex](#) ([Forest](#)< N, Nvalue, M, Mvalue > &F, uint appx)
- void [xdmfPolyvertex](#) (integer my_rank, uint appx)
- void [writeMultiBlock](#) ([Forest](#)< N, Nvalue, M, Mvalue > &F, uint appx)
- void [xdmfMultiBlock](#) ([Forest](#)< N, Nvalue, M, Mvalue > &F, integer comsize, integer my_rank, uint offset, uint appx)
- [~Phdf5](#) ()

Private Attributes

- uint [totalnumber](#)

9.12.1 Detailed Description

```
template<size_t N, typename Nvalue, size_t M, typename Mvalue> class Phdf5< N, Nvalue, M, Mvalue >
```

This Writes out [Tree](#) data in hdf5 format in parallel with *.xmf as metadata suitable for paraview and visit.

9.12.2 Constructor & Destructor Documentation

9.12.2.1 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > Phdf5< N, Nvalue, M, Mvalue >::Phdf5 () [inline]`

9.12.2.2 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > Phdf5< N, Nvalue, M, Mvalue >::~~Phdf5 () [inline]`

9.12.3 Member Function Documentation

9.12.3.1 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Phdf5< N, Nvalue, M, Mvalue >::writeMultiBlock (Forest< N, Nvalue, M, Mvalue > &F, uint appx) [inline]`

writes each element as block and mesh is combination of blocks, appx sets the appendix as string for the output file

<the forest size for each processor

9.12.3.2 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Phdf5< N, Nvalue, M, Mvalue >::writePolyvertex (Forest< N, Nvalue, M, Mvalue > & F, uint appx) [inline]`

writes only the centroids of the mesh, *appx* sets the appendix as string for the output file

<the forest size for each processor

9.12.3.3 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Phdf5< N, Nvalue, M, Mvalue >::xdmfMultiBlock (Forest< N, Nvalue, M, Mvalue > & F, integer comsize, integer my_rank, uint offset, uint appx) [inline]`

9.12.3.4 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Phdf5< N, Nvalue, M, Mvalue >::xdmfPolyvertex (integer my_rank, uint appx) [inline]`

9.12.4 Member Data Documentation

9.12.4.1 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > uint Phdf5< N, Nvalue, M, Mvalue >::totalnumber [private]`

The documentation for this class was generated from the following files:

- [/home/jhasbestan/Morton_Parallel_v0/src/include/phdf5.h](#)
- [/home/jhasbestan/Morton_Parallel_v0/src/phdf5.cpp](#)

9.13 `stl::point` Struct Reference

Structure to hold coordinates of a [point](#).

```
#include <parse_stl.h>
```

Public Member Functions

- [point](#) ()
- [point](#) (float *xp*, float *yp*, float *zp*)

Public Attributes

- float [x](#)
- float [y](#)
- float [z](#)

9.13.1 Detailed Description

Structure to hold coordinates of a [point](#).

9.13.2 Constructor & Destructor Documentation

9.13.2.1 `stl::point::point ()` [`inline`]

z coordinate default constructor

9.13.2.2 `stl::point::point (float xp, float yp, float zp)` [`inline`]

constructor

9.13.3 Member Data Documentation

9.13.3.1 `float stl::point::x`

x coordinate

9.13.3.2 `float stl::point::y`

9.13.3.3 `float stl::point::z`

y coordinate

The documentation for this struct was generated from the following file:

- `/home/jhasbestan/Morton_Parallel_v0/src/include/parse_stl.h`

9.14 `stl::stl_data` Struct Reference

structure to store vector of triangles read in from *.[stl](#) file

```
#include <parse_stl.h>
```

Public Member Functions

- [stl_data](#) (std::string namep)

Public Attributes

- std::string [name](#)
- std::vector< [triangle](#) > [triangles](#)

9.14.1 Detailed Description

structure to store vector of triangles read in from *.[stl](#) file

9.14.2 Constructor & Destructor Documentation

9.14.2.1 `stl::stl_data::stl_data (std::string namep)` `[inline]`

9.14.3 Member Data Documentation

9.14.3.1 `std::string stl::stl_data::name`

9.14.3.2 `std::vector<triangle> stl::stl_data::triangles`

The documentation for this struct was generated from the following file:

- /home/jhasbestan/Morton_Parallel_v0/src/include/[parse_stl.h](#)

9.15 TemplateForest< N, Nvalue, M, Mvalue, T > Class Template Reference

```
#include <templateForest.h>
```

Public Member Functions

- [TemplateForest](#) (T &proc, real *length, real *coords, uint nx, uint ny, uint nz)
- uint [getTotalSize](#) ()
- void [assignSeeds](#) (real *length, T &proc)
- void [assignGeom](#) (T &proc, real *geom_xyz, uint geom_nn)
- void [encodeGeometry](#) ()
- void [refineEachTree](#) (uint nlevel)
- void [moveGeom](#) (T &proc, real *geom_xyz, uint n, real x[3])
- void [getListEachTree](#) ()
- bool [isInSeed](#) (morton< M > &key, uint *counter)
- void [flipAll](#) (morton< N > &key, uint *mylevel, uint *direction)
- void [getDirections](#) (morton< N+M > &key, uint combinedlevel, vector< uint > &directions)
- void [recoverAllZeroSingularity](#) (morton< N+M > &key, const uint &combinedlevel)
- void [combinedLevel](#) (const morton< N+M > &key, uint *level)
- void [findSeedLevelForRcvdMessage](#) (const morton< N+M > &key, uint *mylevel, [Tree](#)< M, Mvalue > &proc)
- void [findSeedLevelForRcvdMessage](#) (const morton< N+M > &key, uint *mylevel, [FullTree](#)< M, Mvalue > &proc)
- void [constructSeedKeyForRcvdMessage](#) (const morton< N+M > &key, const uint &seedlevel, morton< M > &seedkey)
- void [constructElementKeyForRcvdMessage](#) (const morton< N+M > &key, const uint &seedlevel, morton< N > &elementkey)
- void [removeAllZeroSingularity](#) (morton< N+M > &key, const uint &combinedlevel)
- void [getMaxSeedsLevel](#) (T &proc)
- void [findFlipLevel](#) (morton< N+M > key, uint *mylevel, uint *changedirectionlevel, uint *direction)
- void [flipForNbr](#) (morton< N+M > &key, uint *mylevel, uint *changedirectionlevel, uint *direction)
- void [getTotalMeshSize](#) ()
- void [getNbrSeedLevel](#) (morton< N+M > &combinedkey, uint topologylevel, uint *nbrseedleve, [Tree](#)< M, Mvalue > &proc)
- void [getNbrSeedLevel](#) (morton< N+M > &combinedkey, uint topologylevel, uint *nbrseedleve, [FullTree](#)< M, Mvalue > &proc)
- uint [forestsize](#) ()
- void [getElemNbrs](#) ([Tree](#)< M, Mvalue > &proc, const morton< M > key, bitvector< M > &nbr)
- void [comPatternConstruct](#) ([Tree](#)< M, Mvalue > &proc)
- void [comPatternConstruct](#) ([FullTree](#)< M, Mvalue > &proc, vector< uint > &Nbrs)
- void [fourToOneBalance](#) (T &proc)
- void [refineForestBalanced](#) (uint nlevel, T &proc)
- void [nonCollectiveNbrComm](#) ()
- void [refineEachTreeVoxel](#) (uint nlevel)
- [~TemplateForest](#) ()

Protected Attributes

- real [ancestorlength](#) [3]
- real [ancestorcoords](#) [3]
- uint [npx](#)
- uint [npy](#)
- uint [npz](#)

Private Attributes

- [MpiCom](#) Com
- treeList< [Tree](#)< N, Nvalue > > [trees](#)
- bitlist< M > [seeds](#)
- uint [maxseedlevel](#)
- [Tree](#)< M, real > [geom](#)
- vector< uint > [destination](#)
- vector< uint > [sendtag](#)
- vector< uint > [recvtag](#)
- vector< uint > [nbrsOfNbrs](#)
- struct Zoltan_Struct * [zz](#) = nullptr
- [Zoltan_Out](#) zoltan_out
- MPI_Comm [graphComm](#)

Friends

- class [Phdf5](#)

`template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> class TemplateForest< N, Nvalue, M, Mvalue, T >`

9.15.1 Constructor & Destructor Documentation

9.15.1.1 `template<size_t N, typename Nvalue , size_t M, typename Mvalue , class T >
TemplateForest< N, Nvalue, M, Mvalue, T >::TemplateForest (T & proc, real * length,
real * coords, uint nx, uint ny, uint nz) [inline]`

constructor

<part I, initialize the ancestor coords and length, just like we did for class tree

9.15.1.2 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T>
TemplateForest< N, Nvalue, M, Mvalue, T >::~~TemplateForest () [inline]`

9.15.2 Member Function Documentation

9.15.2.1 `template<size_t N, typename Nvalue , size_t M, typename Mvalue , class T > void
TemplateForest< N, Nvalue, M, Mvalue, T >::assignGeom (T & proc, real * geom_xyz,
uint geom_nn) [inline]`

9.15.2.2 `template<size_t N, typename Nvalue , size_t M, typename Mvalue , class T > void
TemplateForest< N, Nvalue, M, Mvalue, T >::assignSeeds (real * length, T & proc)
[inline]`

9.15.2.3 `template<size_t N, typename Nvalue , size_t M, typename Mvalue , class T > void
TemplateForest< N, Nvalue, M, Mvalue, T >::combinedLevel (const morton< N+M > &
key, uint * level) [inline]`

to prevent unnecessary bit operation, the morton code is placed from starting from left hand side

9.15.2.4 `template<size_t N, typename Nvalue , size_t M, typename Mvalue , class T > void
TemplateForest< N, Nvalue, M, Mvalue, T >::comPatternConstruct (FullTree< M,
Mvalue > & proc, vector< uint > & Nbrs) [inline]`

9.15.2.5 `template<size_t N, typename Nvalue , size_t M, typename Mvalue , class T > void
TemplateForest< N, Nvalue, M, Mvalue, T >::comPatternConstruct (Tree< M, Mvalue
> & proc) [inline]`

9.15.2.6 `template<size_t N, typename Nvalue , size_t M, typename Mvalue , class T > void
TemplateForest< N, Nvalue, M, Mvalue, T >::constructElementKeyForRcvdMessage
(const morton< N+M > & key, const uint & seedlevel, morton< N > & elementkey)
[inline]`

9.15.2.7 `template<size_t N, typename Nvalue , size_t M, typename Mvalue , class T > void
TemplateForest< N, Nvalue, M, Mvalue, T >::constructSeedKeyForRcvdMessage (const
morton< N+M > & key, const uint & seedlevel, morton< M > & seedkey) [inline]`

9.15.2.8 `template<size_t N, typename Nvalue , size_t M, typename Mvalue , class T > void
TemplateForest< N, Nvalue, M, Mvalue, T >::encodeGeometry () [inline]`

9.15.2.9 `template<size_t N, typename Nvalue , size_t M, typename Mvalue , class T > void
TemplateForest< N, Nvalue, M, Mvalue, T >::findFlipLevel (morton< N+M > key, uint
* mylevel, uint * changedirectionlevel, uint * direction) [inline]`

same as the function defined in class three except that it workd on (M+N) bits

- 9.15.2.10** `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> void
TemplateForest< N, Nvalue, M, Mvalue, T>::findSeedLevelForRcvdMessage (const
morton< N+M> &key, uint * mylevel, FullTree< M, Mvalue> &proc) [inline]`
- 9.15.2.11** `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> void
TemplateForest< N, Nvalue, M, Mvalue, T>::findSeedLevelForRcvdMessage (const
morton< N+M> &key, uint * mylevel, Tree< M, Mvalue> &proc) [inline]`
- 9.15.2.12** `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> void
TemplateForest< N, Nvalue, M, Mvalue, T>::flipAll (morton< N> &key, uint *
mylevel, uint * direction) [inline]`
- 9.15.2.13** `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> void
TemplateForest< N, Nvalue, M, Mvalue, T>::flipForNbr (morton< N+M> &key, uint
* mylevel, uint * changedirectionlevel, uint * direction) [inline]`

same as the function defined in class three except that it workd on (M+N) bits

- 9.15.2.14** `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> uint
TemplateForest< N, Nvalue, M, Mvalue, T>::forestsize () [inline]`
- 9.15.2.15** `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> void
TemplateForest< N, Nvalue, M, Mvalue, T>::fourToOneBalance (T &proc)
[inline]`

4:1 balance enforced at forest including the other processors

- 9.15.2.16** `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> void
TemplateForest< N, Nvalue, M, Mvalue, T>::getDirections (morton< N+M> &key,
uint combinedlevel, vector< uint> &directions) [inline]`
- 9.15.2.17** `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> void
TemplateForest< N, Nvalue, M, Mvalue, T>::getElemNbrs (Tree< M, Mvalue> &
proc, const morton< M> key, bitvector< M> &nbr) [inline]`

collects the neighbors of a given element

- 9.15.2.18 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T > void TemplateForest< N, Nvalue, M, Mvalue, T >::getListEachTree () [inline]`
- 9.15.2.19 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T > void TemplateForest< N, Nvalue, M, Mvalue, T >::getMaxSeedsLevel (T & proc) [inline]`
- 9.15.2.20 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T > void TemplateForest< N, Nvalue, M, Mvalue, T >::getNbrSeedLevel (morton< N+M > & combinedkey, uint topologylevel, uint * nbrseedleve, FullTree< M, Mvalue > & proc) [inline]`
- 9.15.2.21 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T > void TemplateForest< N, Nvalue, M, Mvalue, T >::getNbrSeedLevel (morton< N+M > & combinedkey, uint topologylevel, uint * nbrseedleve, Tree< M, Mvalue > & proc) [inline]`
- 9.15.2.22 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T > void TemplateForest< N, Nvalue, M, Mvalue, T >::getTotalMeshSize () [inline]`
- 9.15.2.23 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T > uint TemplateForest< N, Nvalue, M, Mvalue, T >::getTotalSize () [inline]`
- 9.15.2.24 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T > bool TemplateForest< N, Nvalue, M, Mvalue, T >::isInSeed (morton< M > & key, uint * counter) [inline]`

check and see if the forest incldues the particluar seed

- 9.15.2.25 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T > void TemplateForest< N, Nvalue, M, Mvalue, T >::moveGeom (T & proc, real * geom_xyz, uint n, real x[3]) [inline]`

moves the geomerty with displacements specified in *x*[3] in x,y and z directions

- 9.15.2.26 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T > void TemplateForest< N, Nvalue, M, Mvalue, T >::nonCollectiveNbrComm () [inline]`
- 9.15.2.27 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T > void TemplateForest< N, Nvalue, M, Mvalue, T >::recoverAllZeroSingularity (morton< N+M > & key, const uint & combinedlevel) [inline]`

note that this function operates on the element key, this is done to remove redundant calc

- 9.15.2.28 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T > void TemplateForest< N, Nvalue, M, Mvalue, T >::refineEachTree (uint nlevel) [inline]`

Refines every [Tree](#) in the list, *nlevels*, balance is satisfired for each tree but not in the global scope

9.15.2.29 `template<size_t N, typename Nvalue , size_t M, typename Mvalue , class T > void
TemplateForest< N, Nvalue, M, Mvalue, T >::refineEachTreeVoxel (uint nlevel)
[inline]`

Refines every [Tree](#) in the list, nlevels, balance is satisfied for each tree but not in the boundaries with other trees

9.15.2.30 `template<size_t N, typename Nvalue , size_t M, typename Mvalue , class T > void
TemplateForest< N, Nvalue, M, Mvalue, T >::refineForestBalanced (uint nlevel, T &
proc) [inline]`

9.15.2.31 `template<size_t N, typename Nvalue , size_t M, typename Mvalue , class T > void
TemplateForest< N, Nvalue, M, Mvalue, T >::removeAllZeroSingularity (morton<
N+M > & key, const uint & combinedlevel) [inline]`

9.15.3 Friends And Related Function Documentation

9.15.3.1 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> friend class
Phdf5 [friend]`

9.15.4 Member Data Documentation

9.15.4.1 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> real
TemplateForest< N, Nvalue, M, Mvalue, T >::ancestorcoords[3] [protected]`

centeroid of the of the first generation (root) element

9.15.4.2 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> real
TemplateForest< N, Nvalue, M, Mvalue, T >::ancestorlength[3] [protected]`

original length of the first generation (root) element

9.15.4.3 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> MPI_Comm
TemplateForest< N, Nvalue, M, Mvalue, T >::Com [private]`

9.15.4.4 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> vector<uint>
TemplateForest< N, Nvalue, M, Mvalue, T >::destination [private]`

9.15.4.5 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> Tree<M,
real> TemplateForest< N, Nvalue, M, Mvalue, T >::geom [private]`

9.15.4.6 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> MPI_Comm
TemplateForest< N, Nvalue, M, Mvalue, T >::graphComm [private]`

Each seed will contain its own geometry points to search, this assumption implicitly coincides processor topology with geometry voxelization this way tree's root will be morton code used in geometry voxelization and no extra operation is necessary

9.15.4.7 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> uint
TemplateForest< N, Nvalue, M, Mvalue, T >::maxseedlevel [private]`

finds the maximum level of the seeds

9.15.4.8 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> vector<uint>
TemplateForest< N, Nvalue, M, Mvalue, T >::nbrsOfNbrs [private]`

9.15.4.9 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> uint
TemplateForest< N, Nvalue, M, Mvalue, T >::npx [protected]`

discretization in x direction, this value for proc tree is 2, therefore forest needs its own value of npz

9.15.4.10 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> uint
TemplateForest< N, Nvalue, M, Mvalue, T >::npz [protected]`

discretization in y direction

9.15.4.11 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> uint
TemplateForest< N, Nvalue, M, Mvalue, T >::npz [protected]`

discretization in z direction

9.15.4.12 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T>
vector<uint> TemplateForest< N, Nvalue, M, Mvalue, T >::recvtag [private]`

9.15.4.13 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> bitlist<M>
TemplateForest< N, Nvalue, M, Mvalue, T >::seeds [private]`

seeds: morton code for boxes to grow tree

9.15.4.14 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T>
vector<uint> TemplateForest< N, Nvalue, M, Mvalue, T >::sendtag [private]`

9.15.4.15 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T>
treeList<Tree<N,Nvalue> > TemplateForest< N, Nvalue, M, Mvalue, T >::trees
[private]`

list of trees that each processor includes

9.15.4.16 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> Zoltan_Out
TemplateForest< N, Nvalue, M, Mvalue, T >::zoltan_out [private]`

9.15.4.17 `template<size_t N, typename Nvalue, size_t M, typename Mvalue, class T> struct
Zoltan_Struct* TemplateForest< N, Nvalue, M, Mvalue, T >::zz = nullptr [read,
private]`

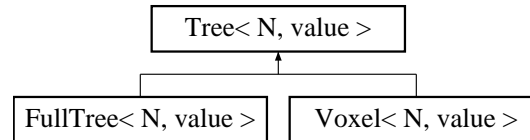
The documentation for this class was generated from the following files:

- /home/jhasbestan/Morton_Parallel_v0/src/include/templateForest.h
- /home/jhasbestan/Morton_Parallel_v0/src/templateForest.cpp

9.16 Tree< N, value > Class Template Reference

This Class Generates a 4:1 balancerd AMR mesh.

#include <tree.h> Inheritance diagram for Tree< N, value >::



Public Member Functions

- [Tree](#) (real *length, real *coords, uint nx, uint ny, uint nz)
- [Tree](#) (real *length, real *coords)
- [Tree](#) ()
- void [construct](#) (real *length, real *coords, uint nx, uint ny, uint nz)
- virtual void [level](#) (morton< N > key, uint *level)
- void [centroid](#) (morton< N > key, real *xyz)
- void [enclosingBox](#) (morton< N > key, real *X)
- virtual bitmap< N, value >::iterator [begin](#) ()
- virtual bitmap< N, value >::iterator [end](#) ()
- virtual uint [size](#) ()
- void [reserve](#) (uint *reservedsize)
- void [siblings](#) (morton< N > key, uint mylevel, morton< N > *sibkey)
- void [refine](#) (morton< N > key)
- void [derefine](#) (morton< N > key)
- void [refineRefineList](#) ()
- void [refineRefineList](#) (bitvector< N > &V)
- void [fourToOneP](#) (uint istart, uint iend)
- void [refineRefineList](#) (uint istart, uint iend)
- void [fourToOne](#) ()
- void [findFlipLevel](#) (morton< N > key, uint *mylevel, uint *changedirectionlevel, uint *direction)
- void [flipForNbr](#) (morton< N > *key, uint *mylevel, uint *changedirectionlevel, uint *direction)
- uint [IsInVectorList](#) (morton< N > key)
- void [addToList](#) (morton< N > key)
- uint [count](#) (morton< N > key)
- void [addToDerefineList](#) (morton< N > key)

If any of the siblings are listed in the dereffinement do not add to the list as derefining one child means removing all the siblings.

- void [derefineDerefineList](#) (uint nlevel)
- uint [isInsideSolid](#) (const morton< N > key, const real *geom_xyz, uint n)
- virtual bitmap< N, value >::iterator [find](#) (morton< N > key)
- void [insertKey](#) (morton< N > key)
- void [convertStl2Morton](#) (uint geom_size, real *geom_xyz)
- void [pushToRefinelist](#) (uint level)
- bool [isBoundary](#) (morton< N > &key)
- void [extractBoundary](#) ()

- void [getDirections](#) (morton< N > &key, vector< uint > &directions)
- uint [refineListSize](#) ()
- void [clearRefineList](#) ()
- void [extractBoundaryP](#) (uint istart, uint iend)
- bool [isInMeshList](#) (const morton< N > &key)
- bool [isInRefineList](#) (const morton< N > &key)
- void [constructHigherLevelNbrs](#) (const morton< N > &key, const uint &keylevel, const uint &direction, morton< N > *nbr)
- void [printMesh](#) ()
- bool [isBoundary](#) (const morton< N > &key, uint direction)
- std::pair< morton< N >, int > [readRefineList](#) (typename std::unordered_map< morton< N >, int >::iterator it)
- morton< N > [readDerefineList](#) (typename std::unordered_map< morton< N >, int >::iterator it)
- void [getKey](#) (uint i, morton< N > &key)
- void [clearMortonSTL](#) ()
- void [retainFourToOne](#) ()
- void [removeFromDerefineList](#) (typename std::unordered_map< morton< N >, int >::iterator it)
- unordered_map< morton< N >, int >::iterator [Dbegin](#) ()
- unordered_map< morton< N >, int >::iterator [Dend](#) ()
- void [derefineDerefineList](#) ()
- void [clearMesh](#) ()
- void [pushToDerefinelist](#) (uint nlevel)
- unordered_map< morton< N >, int >::iterator [Rbegin](#) ()
- unordered_map< morton< N >, int >::iterator [Rend](#) ()
- std::unordered_map< morton< N >, int >::iterator [findInDerefine](#) (morton< N > key)
- void [mortonSTLclear](#) ()
- void [flipRefineElemTag](#) (typename std::unordered_map< morton< N >, int >::iterator it)
- void [refinelistReset](#) ()
- void [constructNonlocalHigherLevelNbrs](#) (const morton< N > &key, const uint &keylevel, const uint &direction, morton< N > *nbr)
- void [insertSeed](#) (morton< N > &key)
- void [insertNbrs](#) (vector< int > &Nbrs)
- void [enclosingBoxFixedLevel](#) (morton< N > key, uint mylevel, real *X)
- void [centroidFixedLevel](#) (morton< N > key, const uint mylevel, real *xyz)
- virtual void [convertCoordToMorton](#) (real *xyz, morton< N > &key)
- std::unordered_map< morton< N >, int >::iterator [findInList](#) (morton< N > key)
- [~Tree](#) ()

Public Attributes

- bitvector< N > [boundarylist](#)

Protected Attributes

- real [ancestorlength](#) [3]
- real [ancestorcoords](#) [3]
- morton< N > [ancestorkey](#) = 0
- uint [npx](#)
- uint [npz](#)
- uint [npz](#)

Private Attributes

- `bitmap< N, value >` [mesh](#)
- `std::unordered_map< morton< N >, int >` [refinelist](#)
- `bitvector< N >` [mortonSTL](#)
- `std::unordered_map< morton< N >, int >` [derefineList](#)

Friends

- class [Hdf5Xmf](#)

9.16.1 Detailed Description

`template<size_t N, typename value> class Tree< N, value >`

This Class Generates a 4:1 balancerd AMR mesh.

9.16.2 Constructor & Destructor Documentation

9.16.2.1 `template<size_t N, typename value > Tree< N, value >::Tree (real * length, real * coords, uint nx, uint ny, uint nz) [inline]`

constructor

9.16.2.2 `template<size_t N, typename value > Tree< N, value >::Tree (real * length, real * coords) [inline]`

constructor

9.16.2.3 `template<size_t N, typename value> Tree< N, value >::Tree () [inline]`

9.16.2.4 `template<size_t N, typename value > Tree< N, value >::~~Tree () [inline]`

Destructor of the class, it frees the memeory pointed by pointer in the hashmap value if allocated

9.16.3 Member Function Documentation

9.16.3.1 `template<size_t N, typename value > void Tree< N, value >::addToDerefineList (morton< N > key) [inline]`

If any of the siblings are listed in the dereffinement do not add to the list as derefining one child means removing all the siblings. adds the element to derefinelist

9.16.3.2 `template<size_t N, typename value > void Tree< N, value >::addToList (morton< N > key) [inline]`

adds element to refinelist

9.16.3.3 `template<size_t N, typename value > bitmap< N, value >::iterator Tree< N, value >::begin () [inline, virtual]`

iterator returning the first object

Reimplemented in [FullTree< N, value >](#).

9.16.3.4 `template<size_t N, typename value > void Tree< N, value >::centroid (morton< N > key, real *xyz) [inline]`

calculates the centroid of the cube given the morton key of the element

9.16.3.5 `template<size_t N, typename value > void Tree< N, value >::centroidFixedLevel (morton< N > key, const uint mylevel, real *xyz) [inline]`

9.16.3.6 `template<size_t N, typename value > void Tree< N, value >::clearMesh () [inline]`

9.16.3.7 `template<size_t N, typename value > void Tree< N, value >::clearMortonSTL () [inline]`

9.16.3.8 `template<size_t N, typename value > void Tree< N, value >::clearRefineList () [inline]`

9.16.3.9 `template<size_t N, typename value > void Tree< N, value >::construct (real * length, real * coords, uint nx, uint ny, uint nz) [inline]`

need to initialize inside forest

9.16.3.10 `template<size_t N, typename value > void Tree< N, value >::constructHigherLevelNbrs (const morton< N > & key, const uint & keylevel, const uint & direction, morton< N > * nbr) [inline]`

9.16.3.11 `template<size_t N, typename value > void Tree< N, value >::constructNonlocalHigherLevelNbrs (const morton< N > & key, const uint & keylevel, const uint & direction, morton< N > * nbr) [inline]`

9.16.3.12 `template<size_t N, typename value > void Tree< N, value >::convertCoordToMorton (real * xyz, morton< N > & key) [inline, virtual]`

converts coordinates of a point to morton code

!< this function is to find a value given the key

Reimplemented in [FullTree< N, value >](#).

9.16.3.13 `template<size_t N, typename value > void Tree< N, value >::convertStl2Morton (uint geom_size, real * geom_xyz) [inline]`

converts [stl](#) coordinates to morton and puts them in morton STL

!< this function is to find a value given the key

9.16.3.14 `template<size_t N, typename value > uint Tree< N, value >::count (morton< N > key) [inline]`

counts the number of elements

9.16.3.15 `template<size_t N, typename value > unordered_map< morton< N >, int >::iterator Tree< N, value >::Dbegin () [inline]`

9.16.3.16 `template<size_t N, typename value > unordered_map< morton< N >, int >::iterator Tree< N, value >::Dend () [inline]`

9.16.3.17 `template<size_t N, typename value > void Tree< N, value >::derefine (morton< N > key) [inline]`

if the morton code does not exist in mesh, refinement is not permitted (derefining a nonexisting element not permitted) Also, if any of the siblings have a higher level of refinement, derefinement is ignored

< if the key does not exist simply ignore doing anything

9.16.3.18 `template<size_t N, typename value > void Tree< N, value >::derefineDerefineList () [inline]`

9.16.3.19 `template<size_t N, typename value> void Tree< N, value >::derefineDerefineList (uint nlevel)`

Derefines the mesh

9.16.3.20 `template<size_t N, typename value > void Tree< N, value >::enclosingBox (morton< N > key, real * X) [inline]`

calculates the range that an element occupies in 3D space for a given Element

9.16.3.21 `template<size_t N, typename value > void Tree< N, value >::enclosingBoxFixedLevel (morton< N > key, uint mylevel, real * X) [inline]`

9.16.3.22 `template<size_t N, typename value > bitmap< N, value >::iterator Tree< N, value >::end () [inline, virtual]`

iterator returning the last object

Reimplemented in [FullTree< N, value >](#).

9.16.3.23 `template<size_t N, typename value > void Tree< N, value >::extractBoundary ()`
`[inline]`

9.16.3.24 `template<size_t N, typename value > void Tree< N, value >::extractBoundaryP (uint`
`istart, uint iend) [inline]`

9.16.3.25 `template<size_t N, typename value > bitmap< N, value >::iterator Tree< N, value`
`>::find (morton< N > key) [inline, virtual]`

this function is to find a value given the key

!< this function is to find a value given the key

Reimplemented in [FullTree< N, value >](#).

9.16.3.26 `template<size_t N, typename value > void Tree< N, value >::findFlipLevel (morton< N`
`> key, uint * mylevel, uint * changedirectionlevel, uint * direction) [inline]`

detects the flip level, this info used in finding nonlocal neighbors

9.16.3.27 `template<size_t N, typename value > std::unordered_map< morton< N >, int`
`>::iterator Tree< N, value >::findInDerefine (morton< N > key) [inline]`

Get the iterator from derefinelist

Parameters:

key E;imate from derefinelist

9.16.3.28 `template<size_t N, typename value > std::unordered_map< morton< N >, int`
`>::iterator Tree< N, value >::findInList (morton< N > key) [inline]`

!< this function is to find a value given the key

9.16.3.29 `template<size_t N, typename value > void Tree< N, value >::flipForNbr (morton< N >`
`* key, uint * mylevel, uint * changedirectionlevel, uint * direction) [inline]`

perform the actual operation to identify the nonlocal nbr

9.16.3.30 `template<size_t N, typename value > void Tree< N, value >::flipRefineElemTag`
`(typename std::unordered_map< morton< N >, int >::iterator it) [inline]`

9.16.3.31 `template<size_t N, typename value > void Tree< N, value >::fourToOne () [inline]`

imposes 4:1 balance given the list of elements to be refined in the vector refine list

< all we are interested is the nonlocal neighbors, i.e. the neighbors of the parents as siblings will have same level

< this approach eliminates search algorithm as now we do not have the restrictions on cutting the cube that we had in the previous approach

9.16.3.32 `template<size_t N, typename value> void Tree< N, value >::fourToOneP (uint istart,
uint iend)`

imposes 4:1 balance locally for each tree, while loop is eliminated due to parallel implementation

9.16.3.33 `template<size_t N, typename value > void Tree< N, value >::getDirections (morton< N
> & key, vector< uint > & directions) [inline]`

9.16.3.34 `template<size_t N, typename value > void Tree< N, value >::getKey (uint i, morton<
N > & key) [inline]`

9.16.3.35 `template<size_t N, typename value > void Tree< N, value >::insertKey (morton< N >
key) [inline]`

Reimplemented in [FullTree< N, value >](#).

9.16.3.36 `template<size_t N, typename value > void Tree< N, value >::insertNbrs (vector< int >
& Nbrs) [inline]`

Parameters:

Nbrs add neighbors

9.16.3.37 `template<size_t N, typename value > void Tree< N, value >::insertSeed (morton< N >
& key) [inline]`

Parameters:

key Eliminate from derefinelist

9.16.3.38 `template<size_t N, typename value > bool Tree< N, value >::isBoundary (const
morton< N > & key, uint direction) [inline]`

9.16.3.39 `template<size_t N, typename value > bool Tree< N, value >::isBoundary (morton< N
> & key) [inline]`

9.16.3.40 `template<size_t N, typename value > bool Tree< N, value >::isInMeshList (const
morton< N > & key) [inline]`

9.16.3.41 `template<size_t N, typename value > bool Tree< N, value >::isInRefineList (const
morton< N > & key) [inline]`

9.16.3.42 `template<size_t N, typename value > uint Tree< N, value >::isInsideSolid (const
morton< N > key, const real * geom_xyz, uint n) [inline]`

tags the elements if any points of the gemoetry resides in the enclosing box

9.16.3.43 `template<size_t N, typename value > uint Tree< N, value >::IsInVectorList (morton< N > key) [inline]`

checks to see if a given code is already in the list

9.16.3.44 `template<size_t N, typename value > void Tree< N, value >::level (morton< N > key, uint * level) [inline, virtual]`

obtains the level of the element from morton key

to prevent unnecessary bit operation, the morton code is placed from starting from left hand side

now look and see if any siblings exist

Reimplemented in [FullTree< N, value >](#).

9.16.3.45 `template<size_t N, typename value > void Tree< N, value >::mortonSTLclear () [inline]`

9.16.3.46 `template<size_t N, typename value > void Tree< N, value >::printMesh () [inline]`

9.16.3.47 `template<size_t N, typename value > void Tree< N, value >::pushToDerefinelist (uint nlevel) [inline]`

!< this function is to find a value given the key

9.16.3.48 `template<size_t N, typename value > void Tree< N, value >::pushToRefinelist (uint nlevel) [inline]`

Note that for dynamic mesh we need to make sure the element exists before adding to this list

!< this function is to find a value given the key

9.16.3.49 `template<size_t N, typename value > unordered_map< morton< N >, int >::iterator Tree< N, value >::Rbegin () [inline]`

9.16.3.50 `template<size_t N, typename value > morton< N > Tree< N, value >::readDerefineList (typename std::unordered_map< morton< N >, int >::iterator it) [inline]`

9.16.3.51 `template<size_t N, typename value > std::pair< morton< N >, int > Tree< N, value >::readRefineList (typename std::unordered_map< morton< N >, int >::iterator it) [inline]`

9.16.3.52 `template<size_t N, typename value > void Tree< N, value >::refine (morton< N > key) [inline]`

performs refinement for a tagged element given the Morton Key

9.16.3.53 `template<size_t N, typename value > void Tree< N, value >::refinelistReset ()`
[inline]

9.16.3.54 `template<size_t N, typename value > uint Tree< N, value >::refineListSize ()`
[inline]

9.16.3.55 `template<size_t N, typename value > void Tree< N, value >::refineRefineList (uint
istart, uint iend)` [inline]

performs the refinement

9.16.3.56 `template<size_t N, typename value > void Tree< N, value >::refineRefineList
(bitvector< N > & V)` [inline]

performs the refinement

9.16.3.57 `template<size_t N, typename value > void Tree< N, value >::refineRefineList ()`
[inline]

performs derefinement on a single element given a morton key performs the refinement

9.16.3.58 `template<size_t N, typename value > void Tree< N, value >::removeFromDerefineList
(typename std::unordered_map< morton< N >, int >::iterator it)` [inline]

Parameters:

it E;iminate from derefinelist

9.16.3.59 `template<size_t N, typename value > unordered_map< morton< N >, int >::iterator
Tree< N, value >::Rend ()` [inline]

9.16.3.60 `template<size_t N, typename value > void Tree< N, value >::reserve (uint *
reservedsize)` [inline]

this function reserves the memory given the reservedsize of the mesh

9.16.3.61 `template<size_t N, typename value > void Tree< N, value >::retainFourToOne ()`
[inline]

9.16.3.62 `template<size_t N, typename value > void Tree< N, value >::siblings (morton< N >
key, uint mylevel, morton< N > * sibkey)` [inline]

extracts the siblings from morton code

9.16.3.63 `template<size_t N, typename value > uint Tree< N, value >::size ()` [inline,
virtual]

returns the size of the mesh

Reimplemented in [FullTree< N, value >](#).

9.16.4 Friends And Related Function Documentation

9.16.4.1 `template<size_t N, typename value> friend class Hdf5Xmf [friend]`

this is a friend class to write out in hdf5 format

9.16.5 Member Data Documentation

9.16.5.1 `template<size_t N, typename value> real Tree< N, value >::ancestorcoords[3] [protected]`

centeroid of the of the first generation (root) element

9.16.5.2 `template<size_t N, typename value> morton<N> Tree< N, value >::ancestorkey = 0 [protected]`

root value is always set as 00000000000000

9.16.5.3 `template<size_t N, typename value> real Tree< N, value >::ancestorlength[3] [protected]`

original length of the first generation (root) element

9.16.5.4 `template<size_t N, typename value> bitvector<N> Tree< N, value >::boundarylist`

list of elements of refinelist that are boundary elements

9.16.5.5 `template<size_t N, typename value> std::unordered_map<morton<N>, int> Tree< N, value >::derefineList [private]`

list of elements tagged to be removed, due to 4:1 balance this list

Reimplemented in [FullTree< N, value >](#).

9.16.5.6 `template<size_t N, typename value> bitmap<N, value> Tree< N, value >::mesh [private]`

base main container for hashmap

Reimplemented in [Voxel< N, value >](#), and [FullTree< N, value >](#).

9.16.5.7 `template<size_t N, typename value> bitvector<N> Tree< N, value >::mortonSTL [private]`

vector to store geometric points in morton code

Reimplemented in [FullTree< N, value >](#).

9.16.5.8 `template<size_t N, typename value> uint Tree< N, value >::npx` `[protected]`

discretization in x direction

9.16.5.9 `template<size_t N, typename value> uint Tree< N, value >::npy` `[protected]`

discretization in y direction

9.16.5.10 `template<size_t N, typename value> uint Tree< N, value >::npz` `[protected]`

discretization in y direction

9.16.5.11 `template<size_t N, typename value> std::unordered_map<morton<N>, int> Tree< N, value >::refinelist` `[private]`

list of elements tagged to be refined

Reimplemented in [FullTree< N, value >](#).

The documentation for this class was generated from the following files:

- [/home/jhasbestan/Morton_Parallel_v0/src/include/tree.h](#)
- [/home/jhasbestan/Morton_Parallel_v0/src/tree.cpp](#)

9.17 `stl::triangle` Struct Reference

structure to store normals and vertices of a [triangle](#)

```
#include <parse_stl.h>
```

Public Member Functions

- [triangle](#) ([point](#) normalp, [point](#) v1p, [point](#) v2p, [point](#) v3p)

Public Attributes

- [point](#) normal
- [point](#) v1
- [point](#) v2
- [point](#) v3

9.17.1 Detailed Description

structure to store normals and vertices of a [triangle](#)

9.17.2 Constructor & Destructor Documentation

9.17.2.1 `stl::triangle::triangle (point normalp, point v1p, point v2p, point v3p)` [`inline`]

9.17.3 Member Data Documentation

9.17.3.1 `point stl::triangle::normal`

normal

9.17.3.2 `point stl::triangle::v1`

coordinates of the vertex 1

9.17.3.3 `point stl::triangle::v2`

coordinates of the vertex 2

9.17.3.4 `point stl::triangle::v3`

coordinates of the vertex 3

The documentation for this struct was generated from the following file:

- `/home/jhasbestan/Morton_Parallel_v0/src/include/parse_stl.h`

9.18 Vector3 Class Reference

```
#include <parse_stl.h>
```

Public Member Functions

- [Vector3](#) (void)
- [Vector3](#) (float [X](#), float [Y](#), float [Z](#))
- [~Vector3](#) (void)
- float [Length](#) ()
- [Vector3 Normalize](#) ()
- [Vector3 Vectors](#) ()

Public Attributes

- float [X](#)
- float [Y](#)
- float [Z](#)

9.18.1 Constructor & Destructor Documentation

9.18.1.1 [Vector3::Vector3](#) (void)

9.18.1.2 [Vector3::Vector3](#) (float *X*, float *Y*, float *Z*)

9.18.1.3 [Vector3::~~Vector3](#) (void)

9.18.2 Member Function Documentation

9.18.2.1 float [Vector3::Length](#) ()

9.18.2.2 [Vector3 Vectors::Normalize](#) ()

9.18.2.3 [Vector3 Vectors::Vectors](#) ()

9.18.3 Member Data Documentation

9.18.3.1 float [Vector3::X](#)

9.18.3.2 float [Vector3::Y](#)

9.18.3.3 float [Vector3::Z](#)

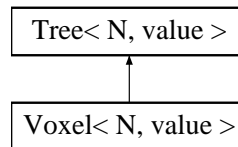
The documentation for this class was generated from the following files:

- [/home/jhasbestan/Morton_Parallel_v0/src/include/parse_stl.h](#)
- [/home/jhasbestan/Morton_Parallel_v0/src/parse_stl.cpp](#)

9.19 Voxel< N, value > Class Template Reference

This Class Generates an unbalanced [Voxel](#) to improve search by geometry partitioning.

#include <tree.h> Inheritance diagram for Voxel< N, value >::



Public Member Functions

- [Voxel](#) (real *length, real *coords)
- void [setLevel](#) (uint *l)
- void [generateSearchTree](#) (real *geom_xyz, uint n)
- void [distributeGeomToLeaves](#) (real *geom_xyz, uint n)
- uint [checkSiblingStatus](#) (morton< N > key, morton< N > *sibkey)
- void [derefineGeomTree](#) ()
- bool [IsInsideSegment](#) (morton< N > key, real *xyz)
- [~Voxel](#) ()

Private Attributes

- uint [maxlevel](#)
- uint [numMax](#)
- bitmap< N, value > [mesh](#)
- bitvector< N > [lookup](#)

Friends

- class [Hdf5XmfV](#)

9.19.1 Detailed Description

template<size_t N, typename value> class Voxel< N, value >

This Class Generates an unbalanced [Voxel](#) to improve search by geometry partitioning.

9.19.2 Constructor & Destructor Documentation

9.19.2.1 template<size_t N, typename value> Voxel< N, value >::Voxel (real * *length*, real * *coords*) [inline]

9.19.2.2 template<size_t N, typename value > Voxel< N, value >::~~Voxel () [inline]

Destructor of this class

9.19.3 Member Function Documentation

9.19.3.1 `template<size_t N, typename value> uint Voxel< N, value >::checkSiblingStatus (morton< N > key, morton< N > * sibkey) [inline]`

Parameters:

sibkey checks to see if siblings include any points and whether they have the same level

9.19.3.2 `template<size_t N, typename value> void Voxel< N, value >::derefineGeomTree () [inline]`

checks to see if siblings include any points and whether they have the same level derefines the tree based on geometry

9.19.3.3 `template<size_t N, typename value> void Voxel< N, value >::distributeGeomToLeaves (real * geom_xyz, uint n) [inline]`

distributes geometry to different cells (leaves)

9.19.3.4 `template<size_t N, typename value> void Voxel< N, value >::generateSearchTree (real * geom_xyz, uint n) [inline]`

generates an initial tree

9.19.3.5 `template<size_t N, typename value> bool Voxel< N, value >::IsInsideSegment (morton< N > key, real * xyz) [inline]`

9.19.3.6 `template<size_t N, typename value> void Voxel< N, value >::setLevel (uint * l) [inline]`

constructor sets the maximum level for refinement

9.19.4 Friends And Related Function Documentation

9.19.4.1 `template<size_t N, typename value> friend class Hdf5XmfV [friend]`

this is a friend class to write out in hdf5 format

9.19.5 Member Data Documentation

9.19.5.1 `template<size_t N, typename value> bitvector<N> Voxel< N, value >::lookup [private]`

9.19.5.2 `template<size_t N, typename value> uint Voxel< N, value >::maxlevel [private]`

maximum level of refinement

9.19.5.3 `template<size_t N, typename value> bitmap<N, value> Voxel< N, value >::mesh` `[private]`

base main container for hashmap

Reimplemented from [Tree< N, value >](#).

9.19.5.4 `template<size_t N, typename value> uint Voxel< N, value >::numMax` `[private]`

number of elements having the highest level

The documentation for this class was generated from the following files:

- [/home/jhasbestan/Morton_Parallel_v0/src/include/tree.h](#)
- [/home/jhasbestan/Morton_Parallel_v0/src/voxel.cpp](#)

9.20 Zoltan_Out Struct Reference

This structure is an interface to store the output from Zoltan.

```
#include <typedefs.h>
```

Public Attributes

- int [changes](#)
- int [numGidEntries](#)
- int [numLidEntries](#)
- int [numImport](#)
- int [numExport](#)
- unsigned int * [importGlobalGids](#)
- unsigned int * [importLocalGids](#)
- unsigned int * [exportGlobalGids](#)
- unsigned int * [exportLocalGids](#)
- int * [importProcs](#)
- int * [importToPart](#)
- int * [exportProcs](#)
- int * [exportToPart](#)
- int * [parts](#)

9.20.1 Detailed Description

This structure is an interface to store the output from Zoltan.

9.20.2 Member Data Documentation

- 9.20.2.1 `int Zoltan_Out::changes`
- 9.20.2.2 `unsigned int* Zoltan_Out::exportGlobalGids`
- 9.20.2.3 `unsigned int* Zoltan_Out::exportLocalGids`
- 9.20.2.4 `int* Zoltan_Out::exportProcs`
- 9.20.2.5 `int* Zoltan_Out::exportToPart`
- 9.20.2.6 `unsigned int* Zoltan_Out::importGlobalGids`
- 9.20.2.7 `unsigned int* Zoltan_Out::importLocalGids`
- 9.20.2.8 `int* Zoltan_Out::importProcs`
- 9.20.2.9 `int* Zoltan_Out::importToPart`
- 9.20.2.10 `int Zoltan_Out::numExport`
- 9.20.2.11 `int Zoltan_Out::numGidEntries`
- 9.20.2.12 `int Zoltan_Out::numImport`
- 9.20.2.13 `int Zoltan_Out::numLidEntries`
- 9.20.2.14 `int* Zoltan_Out::parts`

The documentation for this struct was generated from the following file:

- `/home/jhasbestan/Morton_Parallel_v0/src/include/typedefs.h`

Chapter 10

File Documentation

10.1 `/home/jhasbestan/Morton_Parallel_v0/src/communicate.cpp` File Reference

```
#include "communicate.h"  
#include "datatype.h"
```

Functions

- static MPI_Datatype [ConvertType](#) ([Abstraction::DataType](#) type)

10.1.1 Function Documentation

10.1.1.1 static MPI_Datatype ConvertType ([Abstraction::DataType](#) *type*) `[static]`

communicate class member functions [CommPoint2Point](#) This Class is a wrapper around MPI functions used in this project

Templating the "Intrinsic Type Conversion Using Template Specialization" message tag and mpi_-communicators are assigned by default if user doesnt assign them default tag is 0 and default Communicator is MPI_COMM_WORLD

10.2 /home/jhasbestan/Morton_Parallel_v0/src/forest.cpp File Reference

```
#include "forest.h"  
#include "definitions.h"
```

Defines

- #define [SENDBOOL](#) 1
- #define [METHOD](#) 2
- #define [REORDER](#) 0

10.2.1 Define Documentation

10.2.1.1 #define METHOD 2

10.2.1.2 #define REORDER 0

10.2.1.3 #define SENDBOOL 1

10.3 /home/jhasbestan/Morton_Parallel_v0/src/Ftree_top.cpp File Reference

```
#include "typedefs.h"
#include "communicate.h"
#include "datatype.h"
#include "forest.h"
#include "phdf5.h"
#include "tree.h"
#include "scale.h"
```

Defines

- #define [PROCSIZE](#) 64
- #define [TREESIZE](#) 64

Functions

- void [fTreeProessorTopology](#) (int argc, char *pArgs[])

10.3.1 Define Documentation

10.3.1.1 #define PROCSIZE 64

10.3.1.2 #define TREESIZE 64

10.3.2 Function Documentation

10.3.2.1 void fTreeProessorTopology (int *argc*, char * *pArgs*[])

10.4 `/home/jhasbestan/Morton_Parallel_v0/src/full_tree.cpp` **File Reference**

```
#include "tree.h"  
#include "definitions.h"  
#include "typedefs.h"
```


10.5 /home/jhasbestan/Morton_Parallel_v0/src/include/communicate.h File Reference

```
#include "definitions.h"
```

Classes

- struct [MpiCom](#)
class for embedding data related to the communicator
- struct [Message](#)
struct that embeds data related to the message and envelope
- class [CommPoint2Point< Type >](#)
A template wrapper around MPI functions for point to point communication.
- class [CommCollective< Type >](#)
is a template wrapper around MPI functions for collective communicatios

10.6 /home/jhasbestan/Morton_Parallel_v0/src/include/datatype.h File Reference

```
#include "definitions.h"
```

Namespaces

- namespace [Abstraction](#)

Enumerations

- enum [Abstraction::DataType](#) {
 [Abstraction::type_byte](#), [Abstraction::type_char](#), [Abstraction::type_unsigned_char](#),
 [Abstraction::type_short](#),
 [Abstraction::type_unsigned_short](#), [Abstraction::type_int](#), [Abstraction::type_unsigned_int](#),
 [Abstraction::type_long](#),
 [Abstraction::type_unsigned_long](#), [Abstraction::type_float](#), [Abstraction::type_double](#) }

Functions

- [template<class T >](#)
 [Abstraction::DataType getAbstractionDataType \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< nullptr_t > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< char > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< unsigned char > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< short > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< unsigned short > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< int > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< unsigned int > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< long > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< unsigned long > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< float > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< double > \(\)](#)

10.6.1 Function Documentation

10.6.1.1 `template<class T > Abstraction::DataType getAbstractionDataType () [inline]`

Specilizations for the template class

10.6.1.2 `template<> Abstraction::DataType getAbstractionDataType< char > () [inline]`

10.6.1.3 `template<> Abstraction::DataType getAbstractionDataType< double > () [inline]`

10.6.1.4 `template<> Abstraction::DataType getAbstractionDataType< float > () [inline]`

10.6.1.5 `template<> Abstraction::DataType getAbstractionDataType< int > () [inline]`

10.6.1.6 `template<> Abstraction::DataType getAbstractionDataType< long > () [inline]`

10.6.1.7 `template<> Abstraction::DataType getAbstractionDataType< nullptr_t > ()
[inline]`

10.6.1.8 `template<> Abstraction::DataType getAbstractionDataType< short > () [inline]`

10.6.1.9 `template<> Abstraction::DataType getAbstractionDataType< unsigned char > ()
[inline]`

10.6.1.10 `template<> Abstraction::DataType getAbstractionDataType< unsigned int > ()
[inline]`

10.6.1.11 `template<> Abstraction::DataType getAbstractionDataType< unsigned long > ()
[inline]`

10.6.1.12 `template<> Abstraction::DataType getAbstractionDataType< unsigned short > ()
[inline]`

10.7 /home/jhasbestan/Morton_Parallel_v0/src/include/definitions.h File Reference

```
#include <algorithm>
#include <bitset>
#include <cstdint>
#include <cstdio>
#include <cstdlib>
#include <functional>
#include <iostream>
#include <stack>
#include <unordered_map>
#include <vector>
#include <list>
#include <unistd.h>
#include <mpi.h>
#include <memory>
#include <time.h>
#include <stdexcept>
#include "zoltan.h"
#include <cstddef>
#include <string>
#include <fstream>
#include <sstream>
#include <unordered_set>
#include <utility>
#include <iomanip>
#include <cmath>
```

Defines

- #define [hash](#) 0
- #define [nonnative](#) 1
- #define [WSIZE](#) 3
- #define [RED](#) "\033[01;31m"
- #define [GREEN](#) "\033[22;32m"
- #define [YELLOW](#) "\033[22;33m"
- #define [BLUE](#) "\033[22;34m"
- #define [MAGENTA](#) "\033[22;35m"
- #define [CYAN](#) "\033[22;36m"

- `#define RESET "\033[22;0m"`

10.7.1 Define Documentation

10.7.1.1 `#define BLUE "\033[22;34m"`

10.7.1.2 `#define CYAN "\033[22;36m"`

10.7.1.3 `#define GREEN "\033[22;32m"`

10.7.1.4 `#define hash 0`

10.7.1.5 `#define MAGENTA "\033[22;35m"`

10.7.1.6 `#define nonnative 1`

10.7.1.7 `#define RED "\033[01;31m"`

10.7.1.8 `#define RESET "\033[22;0m"`

10.7.1.9 `#define WSIZE 3`

10.7.1.10 `#define YELLOW "\033[22;33m"`

10.8 /home/jhasbestan/Morton_Parallel_v0/src/include/forest.h File Reference

```
#include "communicate.h"  
#include "definitions.h"  
#include "tree.h"  
#include "typedefs.h"
```

Classes

- class [Forest< N, Nvalue, M, Mvalue >](#)
template class that is a forest of octrees with semi-structured process topology

10.9 /home/jhasbestan/Morton_Parallel_v0/src/include/parse_stl.h File Reference

```
#include <string>
#include <vector>
#include <cassert>
#include <fstream>
#include <iostream>
#include <sstream>
#include <streambuf>
```

Classes

- struct [stl::point](#)
Structure to hold coordinates of a [point](#).
- struct [stl::triangle](#)
structure to store normals and vertices of a [triangle](#)
- struct [stl::stl_data](#)
*structure to store vector of triangles read in from *.[stl](#) file*
- class [Vector3](#)

Namespaces

- namespace [stl](#)

Functions

- [std::ostream & stl::operator<<](#) ([std::ostream &out](#), [const triangle &t](#))
- [stl_data stl::parse_stl](#) ([const std::string &stl_path](#))
- void [checkMesh](#) ([std::vector< stl::triangle > &triangles](#))

10.9.1 Function Documentation

10.9.1.1 void checkMesh (std::vector< stl::triangle > & triangles)

10.10 /home/jhasbestan/Morton_Parallel_v0/src/include/phdf5.h File Reference

```
#include "definitions.h"
```

Classes

- class [Phdf5](#)< N, Nvalue, M, Mvalue >

*This Writes out [Tree](#) data in hdf5 format in parallel with *.xmf as metadata suitable for paraview and visit.*

10.11 /home/jhasbestan/Morton_Parallel_v0/src/include/scale.h File Reference

```
#include "definitions.h"  
#include "typedefs.h"
```

Classes

- class [FullOctreeTop< N >](#)

10.12 /home/jhasbestan/Morton_Parallel_v0/src/include/templateForest.h

File Reference

```
#include "communicate.h"  
#include "definitions.h"  
#include "tree.h"  
#include "typedefs.h"
```

Classes

- class [TemplateForest< N, Nvalue, M, Mvalue, T >](#)

10.13 /home/jhasbestan/Morton_Parallel_v0/src/include/tree.h File Reference

```
#include "definitions.h"
```

Classes

- class [Tree< N, value >](#)

This Class Generates a 4:1 balancerd AMR mesh.

- class [Voxel< N, value >](#)

This Class Generates an unbalancerd [Voxel](#) to improve search by geometry partitioning.

- class [FullTree< N, value >](#)

This Class is specifically designed for weak analysis to operate on the fulltree topology in this class, level is preset by the user and the level function is redefined to make use of the polymorphism.

10.14 /home/jhasbestan/Morton_Parallel_v0/src/include/typedefs.h File Reference

```
#include "definitions.h"
#include "zoltan.h"
```

Classes

- struct [CenterCoords](#)

Stores the coordinate of the centroid of the elements.

- struct [Zoltan_Out](#)

This structure is an interface to store the output from Zoltan.

Defines

- #define [Rma](#) 1
- #define [DEBUG](#) 0

Typedefs

- typedef std::vector< [CenterCoords](#) > [Center_coords](#)

Functions

- bool [zoltanGeometricPartitioner](#) (const uint size, const uint ncube_total, const uint offset, const int method, struct Zoltan_Struct *zz, const [Center_coords](#) &XYZ, real *weight, [Zoltan_Out](#) *zoltan_out)
- bool [zoltanGeometricPartitionerSerial](#) (const uint size, const uint ncube_total, const uint offset, const int method, struct Zoltan_Struct *zz, const [Center_coords](#) &XYZ, real *weight, [Zoltan_Out](#) *zoltan_out, int comsize)
- void [treeProcessorTopology](#) (int argc, char *pArgs[])
- void [fTreeProessorTopology](#) (int argc, char *pArgs[])
- void [readSTLGeom](#) (int argc, char *argv[], real **triangle_center, int *nn, const real *xyz)
- void [TwoPowN](#) (uint b, real *result)

10.14.1 Define Documentation

10.14.1.1 `#define DEBUG 0`

10.14.1.2 `#define Rma 1`

10.14.2 Typedef Documentation

10.14.2.1 `typedef std::vector<CenterCoords> Center_coords`

10.14.3 Function Documentation

10.14.3.1 `void fTreeProessorTopology (int argc, char * pArgs[])`

10.14.3.2 `void readSTLGeom (int argc, char * argv[], real ** triangle_center, int * nn, const real * xyz)`

10.14.3.3 `void treeProcessorTopology (int argc, char * pArgs[])`

10.14.3.4 `void TwoPowN (uint b, real * result) [inline]`

10.14.3.5 `bool zoltanGeometricPartitioner (const uint size, const uint ncube_total, const uint offset, const int method, struct Zoltan_Struct * zz, const Center_coords & XYZ, real * weight, Zoltan_Out * zoltan_out)`

10.14.3.6 `bool zoltanGeometricPartitionerSerial (const uint size, const uint ncube_total, const uint offset, const int method, struct Zoltan_Struct * zz, const Center_coords & XYZ, real * weight, Zoltan_Out * zoltan_out, int comsize)`

10.15 /home/jhasbestan/Morton_Parallel_v0/src/main.cpp File Reference

```
#include "typedefs.h"
#include "communicate.h"
#include "datatype.h"
#include "forest.h"
#include "templateForest.h"
#include "phdf5.h"
#include "tree.h"
#include "scale.h"
```

Defines

- `#define` [PROCSIZE](#) 32
- `#define` [TREESIZE](#) 32
- `#define` [WEAK](#) 2
- `#define` [ZOLTAN_ON](#) 1
- `#define` [WEIGHT](#) 1
- `#define` [ZOLTAN_GEOMETRIC_PARTITION](#) 1

Functions

- `int` [main](#) (int *argc*s, char **pArgs*[])

10.15.1 Define Documentation

10.15.1.1 `#define` [PROCSIZE](#) 32

10.15.1.2 `#define` [TREESIZE](#) 32

10.15.1.3 `#define` [WEAK](#) 2

10.15.1.4 `#define` [WEIGHT](#) 1

10.15.1.5 `#define` [ZOLTAN_GEOMETRIC_PARTITION](#) 1

10.15.1.6 `#define` [ZOLTAN_ON](#) 1

10.15.2 Function Documentation

10.15.2.1 `int` [main](#) (int *argc*s, char **pArgs*[])

10.16 /home/jhasbestan/Morton_Parallel_v0/src/parse_stl.cpp File Reference

```
#include "parse_stl.h"
#include "definitions.h"
```

Namespaces

- namespace [stl](#)

Defines

- #define [MYSCALE](#) 0.5
- #define [CHECK_MESH](#) 0

Functions

- std::ostream & [stl::operator<<](#) (std::ostream &out, const point p)
- std::ostream & [stl::operator<<](#) (std::ostream &out, const triangle &t)
- float [stl::parse_float](#) (std::ifstream &s)
- point [stl::parse_point](#) (std::ifstream &s)
- stl_data [stl::parse_stl](#) (const std::string &stl_path)
- void [readSTLGeom](#) (int argc, char *argv[], real **triangle_center, int *nn, const real *xyz)
- void [checkMesh](#) (std::vector< [stl::triangle](#) > &triangles)

10.16.1 Define Documentation

10.16.1.1 #define [CHECK_MESH](#) 0

10.16.1.2 #define [MYSCALE](#) 0.5

10.16.2 Function Documentation

10.16.2.1 void [checkMesh](#) (std::vector< [stl::triangle](#) > & *triangles*)

10.16.2.2 void [readSTLGeom](#) (int *argc*, char * *argv*[], real ** *triangle_center*, int * *nn*, const real * *xyz*)

10.17 /home/jhasbestan/Morton_Parallel_v0/src/phdf5.cpp File Reference

```
#include "forest.h"
#include "hdf5.h"
#include "phdf5.h"
```

Defines

- #define [H5FILE_NAME](#) "soln/Pxdmf3d%u.h5"
- #define [XDMF_NAME](#) "soln/Pxdmf3d%u.xmf"
- #define [H5FILE](#) "Pxdmf3d%u.h5"
- #define [offset0](#) 156
- #define [offset1](#) 2005

Functions

- static void [integer_string](#) (char *strin, int i)

10.17.1 Define Documentation

10.17.1.1 #define [H5FILE](#) "Pxdmf3d%u.h5"

10.17.1.2 #define [H5FILE_NAME](#) "soln/Pxdmf3d%u.h5"

10.17.1.3 #define [offset0](#) 156

10.17.1.4 #define [offset1](#) 2005

10.17.1.5 #define [XDMF_NAME](#) "soln/Pxdmf3d%u.xmf"

10.17.2 Function Documentation

10.17.2.1 static void [integer_string](#) (char * *strin*, int *i*) [[static](#)]

10.18 /home/jhasbestan/Morton_Parallel_v0/src/scale.cpp File Reference

```
#include "scale.h"
```

10.19 /home/jhasbestan/Morton_Parallel_v0/src/templateForest.cpp

File Reference

```
#include "templateForest.h"  
#include "definitions.h"
```

Defines

- #define [SENDBOOL](#) 1
- #define [METHOD](#) 3
- #define [REORDER](#) 0
- #define [OVERLAP](#) 0

10.19.1 Define Documentation

10.19.1.1 #define METHOD 3

10.19.1.2 #define OVERLAP 0

10.19.1.3 #define REORDER 0

10.19.1.4 #define SENDBOOL 1

10.20 /home/jhasbestan/Morton_Parallel_v0/src/tree.cpp File Reference

```
#include "tree.h"  
#include "definitions.h"  
#include "typedefs.h"
```

10.21 /home/jhasbestan/Morton_Parallel_v0/src/tree_proc.cpp File Reference

```
#include "typedefs.h"
#include "communicate.h"
#include "datatype.h"
#include "forest.h"
#include "phdf5.h"
#include "tree.h"
#include "scale.h"
```

Defines

- #define [PROCSIZE](#) 64
- #define [TREESIZE](#) 64
- #define [ZOLTAN_ON](#) 1
- #define [WEIGHT](#) 1
- #define [ZOLTAN_GEOMETRIC_PARTITION](#) 1

Functions

- void [treeProcessorTopology](#) (int argc, char *pArgs[])

10.21.1 Define Documentation

10.21.1.1 #define [PROCSIZE](#) 64

10.21.1.2 #define [TREESIZE](#) 64

10.21.1.3 #define [WEIGHT](#) 1

10.21.1.4 #define [ZOLTAN_GEOMETRIC_PARTITION](#) 1

10.21.1.5 #define [ZOLTAN_ON](#) 1

10.21.2 Function Documentation

10.21.2.1 void [treeProcessorTopology](#) (int *argc*, char **pArgs*[])

10.22 /home/jhasbestan/Morton_Parallel_v0/src/voxel.cpp File Reference

```
#include "tree.h"
```

10.23 /home/jhasbestan/Morton_Parallel_v0/src/zoltan.cpp File Reference

```
#include <stdlib.h>
#include "typedefs.h"
#include "tree.h"
```

Classes

- struct [MeshData](#)

struct to supply information required by zoltan for geometric partitioners

- struct [GraphData](#)

struct to supply information required by zoltan for geometric partitioners

Defines

- #define [TOL](#) "1.1"

Functions

- static int [get_number_of_objects](#) (void *data, int *ierr)
- static int [get_num_geometry](#) (void *data, int *ierr)
- static void [get_object_list](#) (void *data, int sizeGID, int sizeLID, ZOLTAN_ID_PTR globalID, ZOLTAN_ID_PTR localID, int wgt_dim, float *obj_wgts, int *ierr)
- static void [get_geometry_list](#) (void *data, int sizeGID, int sizeLID, int num_obj, ZOLTAN_ID_PTR globalID, ZOLTAN_ID_PTR localID, int num_dim, double *geom_vec, int *ierr)
- bool [zoltanGeometricPartitioner](#) (const uint size, const uint ncube_total, const uint offset, const int method, struct Zoltan_Struct *zz, const [Center_coords](#) &XYZ, real *weight, [Zoltan_Out](#) *zoltan_out)
- bool [zoltanGeometricPartitionerSerial](#) (const uint size, const uint ncube_total, const uint offset, const int method, struct Zoltan_Struct *zz, const [Center_coords](#) &XYZ, real *weight, [Zoltan_Out](#) *zoltan_out, int comsize)

10.23.1 Define Documentation

10.23.1.1 `#define TOL "1.1"`

10.23.2 Function Documentation

10.23.2.1 `static void get_geometry_list (void * data, int sizeGID, int sizeLID, int num_obj, ZOLTAN_ID_PTR globalID, ZOLTAN_ID_PTR localID, int num_dim, double * geom_vec, int * ierr) [static]`

10.23.2.2 `static int get_num_geometry (void * data, int * ierr) [static]`

10.23.2.3 `static int get_number_of_objects (void * data, int * ierr) [static]`

10.23.2.4 `static void get_object_list (void * data, int sizeGID, int sizeLID, ZOLTAN_ID_PTR globalID, ZOLTAN_ID_PTR localID, int wgt_dim, float * obj_wgts, int * ierr) [static]`

10.23.2.5 `bool zoltanGeometricPartitioner (const uint size, const uint ncube_total, const uint offset, const int method, struct Zoltan_Struct * zz, const Center_coords & XYZ, real * weight, Zoltan_Out * zoltan_out)`

10.23.2.6 `bool zoltanGeometricPartitionerSerial (const uint size, const uint ncube_total, const uint offset, const int method, struct Zoltan_Struct * zz, const Center_coords & XYZ, real * weight, Zoltan_Out * zoltan_out, int comsize)`

Index

- ~CommCollective
 - CommCollective, [19](#)
- ~CommPoint2Point
 - CommPoint2Point, [22](#)
- ~Forest
 - Forest, [27](#)
- ~FullOctreeTop
 - FullOctreeTop, [34](#)
- ~FullTree
 - FullTree, [37](#)
- ~Phdf5
 - Phdf5, [45](#)
- ~TemplateForest
 - TemplateForest, [50](#)
- ~Tree
 - Tree, [59](#)
- ~Vector3
 - Vector3, [69](#)
- ~Voxel
 - Voxel, [70](#)
- /home/jhasbestan/Morton_Parallel_v0/src/ Directory Reference, [14](#)
- /home/jhasbestan/Morton_Parallel_v0/src/Ftree_top.cpp, [77](#)
- /home/jhasbestan/Morton_Parallel_v0/src/communicate.cpp, [75](#)
- /home/jhasbestan/Morton_Parallel_v0/src/forest.cpp, [76](#)
- /home/jhasbestan/Morton_Parallel_v0/src/full_tree.cpp, [78](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/ Directory Reference, [13](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/communicate.h, [79](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/datatype.h, [80](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/definitions.h, [82](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/forest.h, [84](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/parse_stl.h, [85](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/phdf5.h, [86](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/scale.h, [87](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/templateForest.h, [88](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/tree.h, [89](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/typedefs.h, [90](#)
- /home/jhasbestan/Morton_Parallel_v0/src/main.cpp, [92](#)
- /home/jhasbestan/Morton_Parallel_v0/src/parse_stl.cpp, [93](#)
- /home/jhasbestan/Morton_Parallel_v0/src/phdf5.cpp, [94](#)
- /home/jhasbestan/Morton_Parallel_v0/src/scale.cpp, [95](#)
- /home/jhasbestan/Morton_Parallel_v0/src/templateForest.cpp, [96](#)
- /home/jhasbestan/Morton_Parallel_v0/src/tree.cpp, [97](#)
- /home/jhasbestan/Morton_Parallel_v0/src/tree_proc.cpp, [98](#)
- /home/jhasbestan/Morton_Parallel_v0/src/voxel.cpp, [99](#)
- /home/jhasbestan/Morton_Parallel_v0/src/zoltan.cpp, [100](#)
- Abstraction, [15](#)
 - DataType, [15](#)
 - type_byte, [15](#)
 - type_char, [15](#)
 - type_double, [15](#)
 - type_float, [15](#)
 - type_int, [15](#)
 - type_long, [15](#)
 - type_short, [15](#)
 - type_unsigned_char, [15](#)
 - type_unsigned_int, [15](#)
 - type_unsigned_long, [15](#)
 - type_unsigned_short, [15](#)
- addToDerefineList
 - Tree, [59](#)
- addToList
 - Tree, [59](#)
- ancestorcoords

- Forest, 31
- TemplateForest, 54
- Tree, 66
- ancestorkey
 - Tree, 66
- ancestorlength
 - Forest, 31
 - TemplateForest, 54
 - Tree, 66
- assignGeom
 - Forest, 27
 - TemplateForest, 51
- assignProcs
 - FullTree, 37
- assignReciever
 - CommPoint2Point, 22
- assignSeeds
 - Forest, 27
 - TemplateForest, 51
- assignSender
 - CommPoint2Point, 22
- bcast
 - CommCollective, 19
- begin
 - FullTree, 37
 - Tree, 59
- BLUE
 - definitions.h, 83
- boundarylist
 - Tree, 66
- buf
 - Message, 42
- c
 - GraphData, 40
 - MeshData, 41
- Center_coords
 - typedefs.h, 91
- CenterCoords, 17
 - x, 17
 - y, 17
 - z, 17
- centroid
 - Tree, 60
- centroidFixedLevel
 - Tree, 60
- changes
 - Zoltan_Out, 74
- CHECK_MESH
 - parse_stl.cpp, 93
- checkGraphConsistency
 - Forest, 27
 - FullOctreeTop, 34
- checkMesh
 - parse_stl.cpp, 93
 - parse_stl.h, 85
- checkNbrsOfNbrsConsistency
 - Forest, 27
- checkSiblingStatus
 - Voxel, 71
- checkWithNbrs
 - Forest, 27
- checkZoltanPartConsistency
 - Forest, 27
- clearMesh
 - Tree, 60
- clearMortonSTL
 - Tree, 60
- clearRefineList
 - Tree, 60
- Com
 - CommCollective, 19
 - CommPoint2Point, 23
 - Forest, 31
 - TemplateForest, 54
- combinedLevel
 - Forest, 27
 - TemplateForest, 51
- CommCollective, 18
 - ~CommCollective, 19
 - bcast, 19
 - Com, 19
 - CommCollective, 18
 - getTotalNumber, 19
 - Ibcast, 19
 - IgetTotalNumber, 19
 - Msg, 19
 - waitOnRequest, 19
- CommPoint2Point, 21
 - ~CommPoint2Point, 22
 - assignReciever, 22
 - assignSender, 22
 - Com, 23
 - CommPoint2Point, 21, 22
 - getOffset, 22
 - Irecv, 22
 - Isend, 22
 - Msg, 23
 - myRank, 22
 - mySize, 23
 - recv, 23
 - send, 23
- communicate.cpp
 - ConvertType, 75
- comPatternConstruct
 - Forest, 27
 - TemplateForest, 51

- comsize
 - MpiCom, [44](#)
- construct
 - Tree, [60](#)
- constructCommWeak
 - Forest, [28](#)
- constructElementKeyForRcvdMessage
 - Forest, [28](#)
 - TemplateForest, [51](#)
- constructHigherLevelNbrs
 - Tree, [60](#)
- constructNbrProcs
 - FullOctreeTop, [34](#)
- constructNonlocalHigherLevelNbrs
 - Tree, [60](#)
- constructSeedKeyForRcvdMessage
 - Forest, [28](#)
 - TemplateForest, [51](#)
- convertBitsToDouble
 - Forest, [28](#)
- convertCoordToMorton
 - FullTree, [37](#)
 - Tree, [60](#)
- convertDoubleToBits
 - Forest, [28](#)
- convertRank2Bits
 - FullOctreeTop, [35](#)
- convertStl2Morton
 - Tree, [60](#)
- ConvertType
 - communicate.cpp, [75](#)
- count
 - Message, [42](#)
 - Tree, [60](#)
- createCommGraph
 - Forest, [28](#)
- createNbrsOfNbrs
 - Forest, [28](#)
- CYAN
 - definitions.h, [83](#)
- DataType, [24](#)
 - Abstraction, [15](#)
- datatype
 - Message, [42](#)
- datatype.h
 - getAbstractionDataType, [81](#)
 - getAbstractionDataType< char >, [81](#)
 - getAbstractionDataType< double >, [81](#)
 - getAbstractionDataType< float >, [81](#)
 - getAbstractionDataType< int >, [81](#)
 - getAbstractionDataType< long >, [81](#)
 - getAbstractionDataType< nullptr_t >, [81](#)
 - getAbstractionDataType< short >, [81](#)
 - getAbstractionDataType< unsigned char >, [81](#)
 - getAbstractionDataType< unsigned int >, [81](#)
 - getAbstractionDataType< unsigned long >, [81](#)
 - getAbstractionDataType< unsigned short >, [81](#)
- Dbegin
 - Tree, [61](#)
- DEBUG
 - typedefs.h, [91](#)
- debug
 - Forest, [28](#)
- debugDerefine
 - Forest, [28](#)
- definitions.h
 - BLUE, [83](#)
 - CYAN, [83](#)
 - GREEN, [83](#)
 - hash, [83](#)
 - MAGENTA, [83](#)
 - nonnative, [83](#)
 - RED, [83](#)
 - RESET, [83](#)
 - WSIZE, [83](#)
 - YELLOW, [83](#)
- Dend
 - Tree, [61](#)
- derefine
 - Tree, [61](#)
- derefineDerefineList
 - Tree, [61](#)
- derefineGeomTree
 - Voxel, [71](#)
- derefinelist
 - FullTree, [38](#)
 - Tree, [66](#)
- destination
 - Forest, [32](#)
 - TemplateForest, [54](#)
- distributeGeomToLeaves
 - Voxel, [71](#)
- enclosingBox
 - Tree, [61](#)
- enclosingBoxFixedLevel
 - Tree, [61](#)
- encodeGeometry
 - Forest, [28](#)
 - TemplateForest, [51](#)
- end
 - FullTree, [37](#)
 - Tree, [61](#)
- exportGlobalGids
 - Zoltan_Out, [74](#)
- exportLocalGids

- Zoltan_Out, 74
- exportProcs
 - Zoltan_Out, 74
- exportToPart
 - Zoltan_Out, 74
- extractBoundary
 - Tree, 61
- extractBoundaryP
 - Tree, 62
- find
 - FullTree, 37
 - Tree, 62
- findFlipLevel
 - Forest, 28
 - TemplateForest, 51
 - Tree, 62
- findInDerefine
 - Tree, 62
- findInList
 - Tree, 62
- findSeedLevelForRcvdMessage
 - Forest, 28
 - TemplateForest, 51, 52
- fixedlevel
 - FullTree, 38
- flipAll
 - Forest, 29
 - TemplateForest, 52
- flipForNbr
 - Forest, 29
 - TemplateForest, 52
 - Tree, 62
- flipRefineElemTag
 - Tree, 62
- Forest, 25
 - ~Forest, 27
 - ancestorcoords, 31
 - ancestorlength, 31
 - assignGeom, 27
 - assignSeeds, 27
 - checkGraphConsistency, 27
 - checkNbrsOfNbrsConsistency, 27
 - checkWithNbrs, 27
 - checkZoltanPartConsistency, 27
 - Com, 31
 - combinedLevel, 27
 - comPatternConstruct, 27
 - constructCommWeak, 28
 - constructElementKeyForRcvdMessage, 28
 - constructSeedKeyForRcvdMessage, 28
 - convertBitsToDouble, 28
 - convertDoubleToBits, 28
 - createCommGraph, 28
 - createNbrsOfNbrs, 28
 - debug, 28
 - debugDerefine, 28
 - destination, 32
 - encodeGeometry, 28
 - findFlipLevel, 28
 - findSeedLevelForRcvdMessage, 28
 - flipAll, 29
 - flipForNbr, 29
 - Forest, 27
 - forestsize, 29
 - fourToOneBalance, 29
 - geom, 32
 - getDirections, 29
 - getElemNbrs, 29
 - getListEachTree, 29
 - getMaxSeedsLevel, 30
 - getNbrSeedLevel, 30
 - getTotalMeshSize, 30
 - getTotalSize, 30
 - graphComm, 32
 - isInSeed, 30
 - maxseedlevel, 32
 - moveGeom, 30
 - nbrsOfNbrs, 32
 - npx, 32
 - npz, 32
 - npz, 32
 - Phdf5, 31
 - pushToDerefineEachTree, 30
 - rcvrMessageSize, 30
 - recoverAllZeroSingularity, 30
 - refineEachTree, 30
 - refineEachTreeVoxel, 30
 - refineForestBalanced, 31
 - removeAllZeroSingularity, 31
 - retainFourToOneBalance, 31
 - seeds, 32
 - trees, 32
 - zoltan_out, 32
 - zoltanGeomrepart, 31
 - zz, 33
- forest.cpp
 - METHOD, 76
 - REORDER, 76
 - SENDBOOL, 76
- forestsize
 - Forest, 29
 - TemplateForest, 52
- fourToOne
 - Tree, 62
- fourToOneBalance
 - Forest, 29
 - TemplateForest, 52

- fourToOneP
 - Tree, 62
- Ftree_top.cpp
 - fTreeProessorTopology, 77
 - PROCSIZE, 77
 - TREESIZE, 77
- fTreeProessorTopology
 - Ftree_top.cpp, 77
 - typedefs.h, 91
- fullOctreeLevel
 - FullOctreeTop, 35
- FullOctreeTop, 34
 - ~FullOctreeTop, 34
 - checkGraphConsistency, 34
 - constructNbrProcs, 34
 - convertRank2Bits, 35
 - fullOctreeLevel, 35
 - FullOctreeTop, 34
 - isBoundary, 35
 - Nbr, 35
 - readNbrs, 35
 - readRoot, 35
 - rootKey, 35
- FullTree, 36
 - ~FullTree, 37
 - assignProcs, 37
 - begin, 37
 - convertCoordToMorton, 37
 - derefinelist, 38
 - end, 37
 - find, 37
 - fixedlevel, 38
 - FullTree, 37
 - getLevel, 37
 - insertKey, 37
 - isBoundary, 37
 - level, 38
 - mesh, 38
 - mortonSTL, 38
 - nbrsConstrcut, 38
 - refinelist, 39
 - setLevel, 38
 - size, 38
- generateSearchTree
 - Voxel, 71
- geom
 - Forest, 32
 - TemplateForest, 54
- get_geometry_list
 - zoltan.cpp, 101
- get_num_geometry
 - zoltan.cpp, 101
- get_number_of_objects
 - zoltan.cpp, 101
- get_object_list
 - zoltan.cpp, 101
- getAbstractionDataType
 - datatype.h, 81
- getAbstractionDataType< char >
 - datatype.h, 81
- getAbstractionDataType< double >
 - datatype.h, 81
- getAbstractionDataType< float >
 - datatype.h, 81
- getAbstractionDataType< int >
 - datatype.h, 81
- getAbstractionDataType< long >
 - datatype.h, 81
- getAbstractionDataType< nullptr_t >
 - datatype.h, 81
- getAbstractionDataType< short >
 - datatype.h, 81
- getAbstractionDataType< unsigned char >
 - datatype.h, 81
- getAbstractionDataType< unsigned int >
 - datatype.h, 81
- getAbstractionDataType< unsigned long >
 - datatype.h, 81
- getAbstractionDataType< unsigned short >
 - datatype.h, 81
- getDirections
 - Forest, 29
 - TemplateForest, 52
 - Tree, 63
- getElemNbrs
 - Forest, 29
 - TemplateForest, 52
- getKey
 - Tree, 63
- getLevel
 - FullTree, 37
- getListEachTree
 - Forest, 29
 - TemplateForest, 52
- getMaxSeedsLevel
 - Forest, 30
 - TemplateForest, 53
- getNbrSeedLevel
 - Forest, 30
 - TemplateForest, 53
- getOffset
 - CommPoint2Point, 22
- getTotalMeshSize
 - Forest, 30
 - TemplateForest, 53
- getTotalNumber
 - CommCollective, 19

- getTotalSize
 - Forest, 30
 - TemplateForest, 53
- graphComm
 - Forest, 32
 - TemplateForest, 54
- GraphData, 40
 - c, 40
 - nbrGID, 40
 - nbrIndex, 40
 - nbrProc, 40
 - numMyVertices, 40
 - vertexGID, 40
- GREEN
 - definitions.h, 83
- H5FILE
 - phdf5.cpp, 94
- H5FILE_NAME
 - phdf5.cpp, 94
- hash
 - definitions.h, 83
- Hdf5Xmf
 - Tree, 66
- Hdf5XmfV
 - Voxel, 71
- Ibcast
 - CommCollective, 19
- IgetTotalNumber
 - CommCollective, 19
- importGlobalGids
 - Zoltan_Out, 74
- importLocalGids
 - Zoltan_Out, 74
- importProcs
 - Zoltan_Out, 74
- importToPart
 - Zoltan_Out, 74
- insertKey
 - FullTree, 37
 - Tree, 63
- insertNbrs
 - Tree, 63
- insertSeed
 - Tree, 63
- integer_string
 - phdf5.cpp, 94
- Irecv
 - CommPoint2Point, 22
- isBoundary
 - FullOctreeTop, 35
 - FullTree, 37
 - Tree, 63
- Isend
 - CommPoint2Point, 22
- isInMeshList
 - Tree, 63
- isInRefineList
 - Tree, 63
- isInSeed
 - Forest, 30
 - TemplateForest, 53
- IsInsideSegment
 - Voxel, 71
- isInsideSolid
 - Tree, 63
- IsInVectorList
 - Tree, 63
- Length
 - Vector3, 69
- level
 - FullTree, 38
 - Tree, 64
- lookup
 - Voxel, 71
- MAGENTA
 - definitions.h, 83
- main
 - main.cpp, 92
- main.cpp
 - main, 92
 - PROCSIZE, 92
 - TREESIZE, 92
 - WEAK, 92
 - WEIGHT, 92
 - ZOLTAN_GEOMETRIC_PARTITION, 92
 - ZOLTAN_ON, 92
- maxlevel
 - Voxel, 71
- maxseedlevel
 - Forest, 32
 - TemplateForest, 54
- mesh
 - FullTree, 38
 - Tree, 66
 - Voxel, 71
- MeshData, 41
 - c, 41
 - myGlobalIDs, 41
 - numGlobalPoints, 41
 - numMyPoints, 41
 - w, 41
- Message, 42
 - buf, 42
 - count, 42

- datatype, 42
- print, 42
- reciever, 42
- request, 42
- sender, 43
- status, 43
- tag, 43
- METHOD
 - forest.cpp, 76
 - templateForest.cpp, 96
- mortonSTL
 - FullTree, 38
 - Tree, 66
- mortonSTLclear
 - Tree, 64
- moveGeom
 - Forest, 30
 - TemplateForest, 53
- MpiCom, 44
 - comsize, 44
 - MpiCom, 44
 - mpicom, 44
 - myrank, 44
- mpicom
 - MpiCom, 44
- Msg
 - CommCollective, 19
 - CommPoint2Point, 23
- myGlobalIDs
 - MeshData, 41
- myRank
 - CommPoint2Point, 22
- myrank
 - MpiCom, 44
- MYSCALE
 - parse_stl.cpp, 93
- mySize
 - CommPoint2Point, 23
- name
 - stl::stl_data, 48
- Nbr
 - FullOctreeTop, 35
- nbrGID
 - GraphData, 40
- nbrIndex
 - GraphData, 40
- nbrProc
 - GraphData, 40
- nbrsConstrcut
 - FullTree, 38
- nbrsOfNbrs
 - Forest, 32
 - TemplateForest, 55
- nonCollectiveNbrComm
 - TemplateForest, 53
- nonnative
 - definitions.h, 83
- normal
 - stl::triangle, 68
- Normalize
 - Vector3, 69
- npz
 - Forest, 32
 - TemplateForest, 55
 - Tree, 66
- npz
 - Forest, 32
 - TemplateForest, 55
 - Tree, 67
- npz
 - Forest, 32
 - TemplateForest, 55
 - Tree, 67
- numExport
 - Zoltan_Out, 74
- numGidEntries
 - Zoltan_Out, 74
- numGlobalPoints
 - MeshData, 41
- numImport
 - Zoltan_Out, 74
- numLidEntries
 - Zoltan_Out, 74
- numMax
 - Voxel, 72
- numMyPoints
 - MeshData, 41
- numMyVertices
 - GraphData, 40
- offset0
 - phdf5.cpp, 94
- offset1
 - phdf5.cpp, 94
- operator<<
 - stl, 16
- OVERLAP
 - templateForest.cpp, 96
- parse_float
 - stl, 16
- parse_point
 - stl, 16
- parse_stl
 - stl, 16
- parse_stl.cpp
 - CHECK_MESH, 93

- checkMesh, 93
- MYSCALE, 93
- readSTLGeom, 93
- parse_stl.h
 - checkMesh, 85
- parts
 - Zoltan_Out, 74
- Phdf5, 45
 - ~Phdf5, 45
 - Forest, 31
 - Phdf5, 45
 - TemplateForest, 54
 - totalnumber, 46
 - writeMultiBlock, 45
 - writePolyvertex, 45
 - xdmfMultiBlock, 46
 - xdmfPolyvertex, 46
- phdf5.cpp
 - H5FILE, 94
 - H5FILE_NAME, 94
 - integer_string, 94
 - offset0, 94
 - offset1, 94
 - XDMF_NAME, 94
- point
 - stl::point, 47
- print
 - Message, 42
- printMesh
 - Tree, 64
- PROCSIZE
 - Ftree_top.cpp, 77
 - main.cpp, 92
 - tree_proc.cpp, 98
- pushToDerefineEachTree
 - Forest, 30
- pushToDerefinelist
 - Tree, 64
- pushToRefinelist
 - Tree, 64
- Rbegin
 - Tree, 64
- rcvrMessageSize
 - Forest, 30
- readDerefineList
 - Tree, 64
- readNbrs
 - FullOctreeTop, 35
- readRefineList
 - Tree, 64
- readRoot
 - FullOctreeTop, 35
- readSTLGeom
 - parse_stl.cpp, 93
 - typedefs.h, 91
- reciever
 - Message, 42
- recoverAllZeroSingularity
 - Forest, 30
 - TemplateForest, 53
- recv
 - CommPoint2Point, 23
- recvtag
 - TemplateForest, 55
- RED
 - definitions.h, 83
- refine
 - Tree, 64
- refineEachTree
 - Forest, 30
 - TemplateForest, 53
- refineEachTreeVoxel
 - Forest, 30
 - TemplateForest, 53
- refineForestBalanced
 - Forest, 31
 - TemplateForest, 54
- refinelist
 - FullTree, 39
 - Tree, 67
- refinelistReset
 - Tree, 64
- refineListSize
 - Tree, 65
- refineRefineList
 - Tree, 65
- removeAllZeroSingularity
 - Forest, 31
 - TemplateForest, 54
- removeFromDerefineList
 - Tree, 65
- Rend
 - Tree, 65
- REORDER
 - forest.cpp, 76
 - templateForest.cpp, 96
- request
 - Message, 42
- reserve
 - Tree, 65
- RESET
 - definitions.h, 83
- retainFourToOne
 - Tree, 65
- retainFourToOneBalance
 - Forest, 31
- Rma

- typedefs.h, 91
- rootKey
 - FullOctreeTop, 35
- seeds
 - Forest, 32
 - TemplateForest, 55
- send
 - CommPoint2Point, 23
- SENDBOOL
 - forest.cpp, 76
 - templateForest.cpp, 96
- sender
 - Message, 43
- sendtag
 - TemplateForest, 55
- setLevel
 - FullTree, 38
 - Voxel, 71
- siblings
 - Tree, 65
- size
 - FullTree, 38
 - Tree, 65
- status
 - Message, 43
- stl, 16
 - operator<<, 16
 - parse_float, 16
 - parse_point, 16
 - parse_stl, 16
- stl::point, 47
 - point, 47
 - x, 47
 - y, 47
 - z, 47
- stl::stl_data, 48
 - name, 48
 - stl_data, 48
 - triangles, 48
- stl::triangle, 68
 - normal, 68
 - triangle, 68
 - v1, 68
 - v2, 68
 - v3, 68
- stl_data
 - stl::stl_data, 48
- tag
 - Message, 43
- TemplateForest, 49
 - ~TemplateForest, 50
 - ancestorcoords, 54
 - ancestorlength, 54
 - assignGeom, 51
 - assignSeeds, 51
 - Com, 54
 - combinedLevel, 51
 - comPatternConstruct, 51
 - constructElementKeyForRcvdMessage, 51
 - constructSeedKeyForRcvdMessage, 51
 - destination, 54
 - encodeGeometry, 51
 - findFlipLevel, 51
 - findSeedLevelForRcvdMessage, 51, 52
 - flipAll, 52
 - flipForNbr, 52
 - forestsize, 52
 - fourToOneBalance, 52
 - geom, 54
 - getDirections, 52
 - getElemNbrs, 52
 - getListEachTree, 52
 - getMaxSeedsLevel, 53
 - getNbrSeedLevel, 53
 - getTotalMeshSize, 53
 - getTotalSize, 53
 - graphComm, 54
 - isInSeed, 53
 - maxseedlevel, 54
 - moveGeom, 53
 - nbrsOfNbrs, 55
 - nonCollectiveNbrComm, 53
 - npx, 55
 - npz, 55
 - npz, 55
 - Phdf5, 54
 - recoverAllZeroSingularity, 53
 - recvtag, 55
 - refineEachTree, 53
 - refineEachTreeVoxel, 53
 - refineForestBalanced, 54
 - removeAllZeroSingularity, 54
 - seeds, 55
 - sendtag, 55
 - TemplateForest, 50
 - trees, 55
 - zoltan_out, 55
 - zz, 55
- templateForest.cpp
 - METHOD, 96
 - OVERLAP, 96
 - REORDER, 96
 - SENDBOOL, 96
- TOL
 - zoltan.cpp, 101
- totalnumber

- Phdf5, 46
- Tree, 57
 - ~Tree, 59
 - addToDerefineList, 59
 - addToList, 59
 - ancestorcoords, 66
 - ancestorkey, 66
 - ancestorlength, 66
 - begin, 59
 - boundarylist, 66
 - centroid, 60
 - centroidFixedLevel, 60
 - clearMesh, 60
 - clearMortonSTL, 60
 - clearRefineList, 60
 - construct, 60
 - constructHigherLevelNbrs, 60
 - constructNonlocalHigherLevelNbrs, 60
 - convertCoordToMorton, 60
 - convertStl2Morton, 60
 - count, 60
 - Dbegin, 61
 - Dend, 61
 - derefine, 61
 - derefineDerefineList, 61
 - derefinelist, 66
 - enclosingBox, 61
 - enclosingBoxFixedLevel, 61
 - end, 61
 - extractBoundary, 61
 - extractBoundaryP, 62
 - find, 62
 - findFlipLevel, 62
 - findInDerefine, 62
 - findInList, 62
 - flipForNbr, 62
 - flipRefineElemTag, 62
 - fourToOne, 62
 - fourToOneP, 62
 - getDirections, 63
 - getKey, 63
 - Hdf5Xmf, 66
 - insertKey, 63
 - insertNbrs, 63
 - insertSeed, 63
 - isBoundary, 63
 - isInMeshList, 63
 - isInRefineList, 63
 - isInsideSolid, 63
 - IsInVectorList, 63
 - level, 64
 - mesh, 66
 - mortonSTL, 66
 - mortonSTLclear, 64
 - npx, 66
 - npz, 67
 - npz, 67
 - printMesh, 64
 - pushToDerefinelist, 64
 - pushToRefinelist, 64
 - Rbegin, 64
 - readDerefineList, 64
 - readRefineList, 64
 - refine, 64
 - refinelist, 67
 - refinelistReset, 64
 - refineListSize, 65
 - refineRefineList, 65
 - removeFromDerefineList, 65
 - Rend, 65
 - reserve, 65
 - retainFourToOne, 65
 - siblings, 65
 - size, 65
 - Tree, 59
- tree_proc.cpp
 - PROCSIZE, 98
 - treeProcessorTopology, 98
 - TREESIZE, 98
 - WEIGHT, 98
 - ZOLTAN_GEOMETRIC_PARTITION, 98
 - ZOLTAN_ON, 98
- treeProcessorTopology
 - tree_proc.cpp, 98
 - typedefs.h, 91
- trees
 - Forest, 32
 - TemplateForest, 55
- TREESIZE
 - Ftree_top.cpp, 77
 - main.cpp, 92
 - tree_proc.cpp, 98
- triangle
 - stl::triangle, 68
- triangles
 - stl::stl_data, 48
- TwoPowN
 - typedefs.h, 91
- type_byte
 - Abstraction, 15
- type_char
 - Abstraction, 15
- type_double
 - Abstraction, 15
- type_float
 - Abstraction, 15
- type_int
 - Abstraction, 15

- type_long
 - Abstraction, 15
- type_short
 - Abstraction, 15
- type_unsigned_char
 - Abstraction, 15
- type_unsigned_int
 - Abstraction, 15
- type_unsigned_long
 - Abstraction, 15
- type_unsigned_short
 - Abstraction, 15
- typedefs.h
 - Center_coords, 91
 - DEBUG, 91
 - fTreeProcessorTopology, 91
 - readSTLGeom, 91
 - Rma, 91
 - treeProcessorTopology, 91
 - TwoPowN, 91
 - zoltanGeometricPartitioner, 91
 - zoltanGeometricPartitionerSerial, 91
- v1
 - stl::triangle, 68
- v2
 - stl::triangle, 68
- v3
 - stl::triangle, 68
- Vector3, 69
 - ~Vector3, 69
 - Length, 69
 - Normalize, 69
 - Vector3, 69
 - Vectors, 69
 - X, 69
 - Y, 69
 - Z, 69
- Vectors
 - Vector3, 69
- vertexGID
 - GraphData, 40
- Voxel, 70
 - ~Voxel, 70
 - checkSiblingStatus, 71
 - derefineGeomTree, 71
 - distributeGeomToLeaves, 71
 - generateSearchTree, 71
 - Hdf5XmfV, 71
 - IsInsideSegment, 71
 - lookup, 71
 - maxlevel, 71
 - mesh, 71
 - numMax, 72
 - setLevel, 71
 - Voxel, 70
- w
 - MeshData, 41
- waitOnRequest
 - CommCollective, 19
- WEAK
 - main.cpp, 92
- WEIGHT
 - main.cpp, 92
 - tree_proc.cpp, 98
- writeMultiBlock
 - Phdf5, 45
- writePolyvertex
 - Phdf5, 45
- WSIZE
 - definitions.h, 83
- X
 - Vector3, 69
- x
 - CenterCoords, 17
 - stl::point, 47
- XDMF_NAME
 - phdf5.cpp, 94
- xdmfMultiBlock
 - Phdf5, 46
- xdmfPolyvertex
 - Phdf5, 46
- Y
 - Vector3, 69
- y
 - CenterCoords, 17
 - stl::point, 47
- YELLOW
 - definitions.h, 83
- Z
 - Vector3, 69
- z
 - CenterCoords, 17
 - stl::point, 47
- zoltan.cpp
 - get_geometry_list, 101
 - get_num_geometry, 101
 - get_number_of_objects, 101
 - get_object_list, 101
 - TOL, 101
 - zoltanGeometricPartitioner, 101
 - zoltanGeometricPartitionerSerial, 101
- ZOLTAN_GEOMETRIC_PARTITION
 - main.cpp, 92

- tree_proc.cpp, [98](#)
- ZOLTAN_ON
 - main.cpp, [92](#)
 - tree_proc.cpp, [98](#)
- Zoltan_Out, [73](#)
 - changes, [74](#)
 - exportGlobalGids, [74](#)
 - exportLocalGids, [74](#)
 - exportProcs, [74](#)
 - exportToPart, [74](#)
 - importGlobalGids, [74](#)
 - importLocalGids, [74](#)
 - importProcs, [74](#)
 - importToPart, [74](#)
 - numExport, [74](#)
 - numGidEntries, [74](#)
 - numImport, [74](#)
 - numLidEntries, [74](#)
 - parts, [74](#)
- zoltan_out
 - Forest, [32](#)
 - TemplateForest, [55](#)
- zoltanGeometricPartitioner
 - typedefs.h, [91](#)
 - zoltan.cpp, [101](#)
- zoltanGeometricPartitionerSerial
 - typedefs.h, [91](#)
 - zoltan.cpp, [101](#)
- zoltanGeomrepart
 - Forest, [31](#)
- zz
 - Forest, [33](#)
 - TemplateForest, [55](#)