~~Level 1 tasks non in context, e.g. "chol" Aii ???~~

partition nodes fractional {0.8,0.2}

processor group structure

~~Tasks : SYRK~~

~~MatAsm refactor~~

~~Connecting Contexts MatAsm , Cholesky~~

~~Cholesky ctor ( M )~~

~~Glb-> Connect ( MA , C~~ )

Context host policy

to be moved inside the MatAsm and Cholesky

not global

set active host policy(CHP::….)

set group count ( a,b,c)

set host partitions (100%, {a%,1-a%} , {b%,c%,1-b-c} )

Message Contents

~~General tasks~~

~~Task creation in ductteip engine~~

~~Propagation tasks~~

~~Split/combine context string/ids~~

~~Pack /Unpack~~

Data

Listener

~~Data Handles~~

~~Ictx ctor register itself to glb => new id~~

~~Ictx ask for new handle from glb for its data~~

~~glbctx.craetedata()~~

~~Ictx dtor data or not ?~~

Context fan in/out > 1 ?

Clone input data for other context, after gen tasks finished remove it

Complete Framework:

DuctTeip/engine

Data

Check sts, create , clean,find by sts, populate

Listener

CheckSts,clean,findBysts,senddata ,dta\_recvd

Task

~~Add~~,canRun,checksts,data\_recvd,findin list?,match axs, run,

Upgrade data

Engine

Check : Ack recv,data recv, lsnr,outbox,scheduler,task

New, ~~can terminate~~,

Extern interface: create data,axs, task

Init: for all dl in DynLib do Init(dl);

Run:

Check tasks for run

Check mail box

AddTask

Pack /Unpack

Kernel / run tasks

Engine

Send ~~Task~~,Listener,Data,Prop

Received ~~Task~~,Listener,Data,Prop

Serialize ~~Task~~,Listener,Data,~~Prop~~

Deserialize ~~Task~~,Listener,Data,~~Prop~~

~~MailBox~~

~~Send ( buf,len,Tag,dest)~~

~~Check inbox~~

~~Network~~

~~Isend, Irecv, probe,init,finish,get\_rank, get\_size~~

Data Task Listener Propagate

~~Received Buffer,source,tag~~

~~Check outbox inbox~~

~~Network~~

~~MPI~~

Kernels

Link to Shared Mem Libraries:

SuperGlue

BLAS,LAPACK

Fortran interface

dynamic library load

Dynamic load balancing

Iterative tasks

Fault tolerance

Sparse Matrix

Recursive Tasks

Dynamic Task Generation

Events🡪 Actions

**Data**

*Received*:

For all data->listeners :

If (version matches)remove lsnr

For all tasks:

If (data matches)

If (version matches)

Task.data.state = ready

Add task to run-check queue

Raise data-ready event

*Ready*:

For data->listeners:

If (version matches) lsnr.send\_data

Check tasks for run

*Sent to host ‘h’*:

Remove data->listeners

**Listener**

*Received*:

Add to list

Attach to corresponding data

If (data ready) lsnr.send\_data

*Sent*:

Remove from list

**Task**

*Received:*

Add to list

Check task for run

*Sent:*

Remove from list

*Ready:*

Call kernel

*Finished:*

Upgrade out-data version

Remove from list

Actions 🡪 Events

**Data**

*Upgrade version*:

‘Data ready’

**Listener**

*Remove:*

If (me == listener.host)

Upgrade data version

**Task**

*Check for run*

‘task ready’

*Call kernel*

‘task finished’