~~NCI Australia's~~ Jamie's GPU Hackathon

A lightning talk

Jamie Border

August 26th, 2020

Why so slow?

Num Calls	Tree Time	Func Time	Per Call		filter out	% of total
1	30102	4	4	_Dmain		0.013373901
1	30095	0	0	simulate.Simulation simulate.Simulation.__ctor(immutable(char)[])		0
1	30075	28	28	void simulate.Simulation.run()		0.093617306
261	13546	37	0	void simulate.Simulation.applyWFRPolynomialFit()		0.123708583
16965	13135	1210	0	void reconstruction.WFRScheme.WalshFunctionPCFOverElement(finiteVolume.cell.FVCellWFR[], finiteVolume.face.FVInterfaceWFR[], globalConfig.GlobalConfig*, const(bool), finiteVolume		4.045605002
522	10516	1178	2	void plot.plotter.generatePlotData(double[], double[][], immutable(char)[][immutable(char)[]], bool, immutable(char)[])		3.938613795
261	6738	103	0	void simulate.Simulation.writeDataOut!(["gas.rho", "vel", "gas.p", "gas.u"]).writeDataOut(ulong, bool)		0.344377946
32219331	5780	1094	0	pure nothrow @nogc @trusted double std.math.pow!(double, double).pow(double, double)		3.657761878
67520	5070	1545	0	double[][] Walsh.FWT.FWT_xiToXm(ulong, ulong, double, double, double[])		5.165669197
32219331	4686	3697	0	pure nothrow @nogc @safe real std.math.pow!(double, double).pow(double, double).impl(real, real)		12.36082784
17487	4499	73	0	double[] Walsh.PCF.solveCoeffsWithShockDetectionFromPoints(ulong, ulong, double, double, double[], ref bool, ref double, ref double, ref ulong, ref bool, double)		0.24407369
268840	4135	3201	0	@safe void std.file.append!(immutable(char)[]).append(immutable(char)[], const(void[]))		10.70246414
261	3996	64	0	void simulate.Simulation.calculateErrorNorms(const(ulong), bool)		0.213982413
33408	3169	4	0	double[] Walsh.PCF.generatePolyCoeffsFromPoints(ulong, ulong, double, double, double[], double[])		0.013373901
260	2850	2	0	void simulate.Simulation.calculateInviscidFluxesFromTimeLevelWFR(const(ulong))		0.00668695
16900	2661	14	0	void fluxCalculators.WFRFluxCalc.calculateFluxWFR(finiteVolume.cell.FVCellWFR[], finiteVolume.face.FVInterfaceWFR[], const(ulong), const(bool))		0.046808653
137540	2239	67	0	void finiteVolume.face.FVInterfaceWFR.calculateInviscidFluxWFRCorrections()		0.224012839
260	1317	18	0	void simulate.Simulation.reconstructFromTimeLevelWFR(const(ulong))		0.060182554
85007	1056	359	0	double[] Walsh.FWT.FWT_fiToFaFromPoints(const(ulong), const(double), const(double), const(double[]))		1.2003076
140352	995	501	0	@trusted double Walsh.core.fg_n(const(ulong), const(double), const(double), const(double))		1.675081079
23168418	888	888	0	pure nothrow @nogc @trusted real std.math.pow!(real, long).pow(real, long)		2.969005985

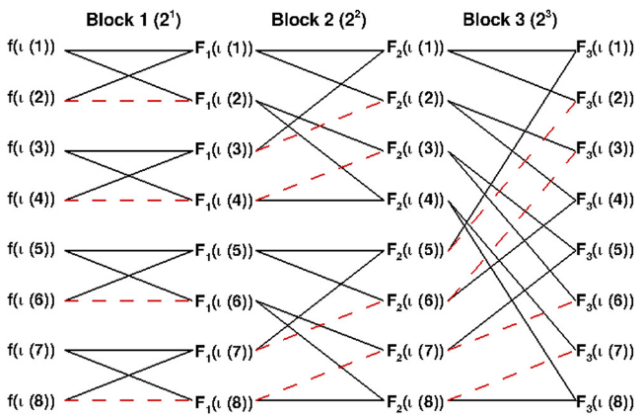
$$f_i = \sum_{n=1}^{2^p} A_n g_n(x_i)$$

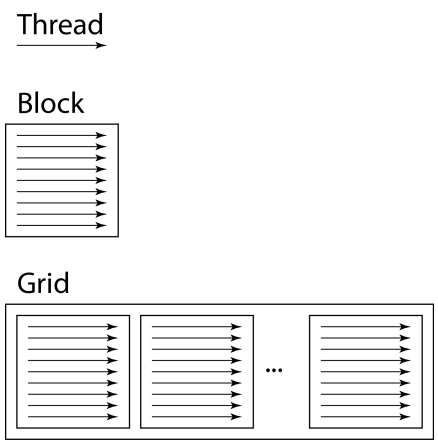
$$x_i^m = \sum_{n=1}^{2^p} B_{n,m} g_n(x_i)$$

with $1 \le i \le 2^p$, $m \ge 0$ and where

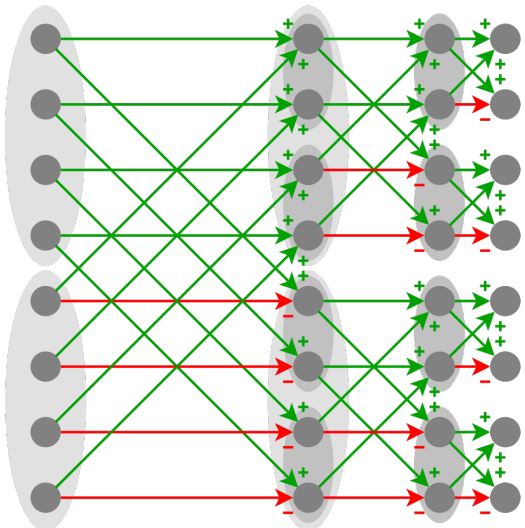
$$A_n = \sum_{i=1}^{2^p} g_n(x_i) f_i \Delta x$$

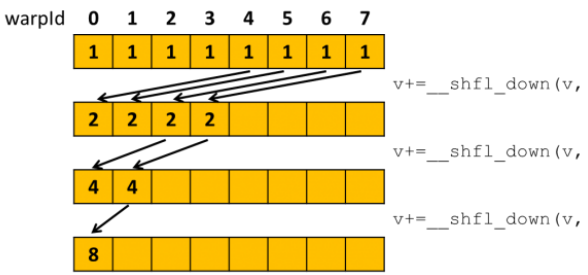
$$B_{n,m} = \sum_{i=1}^{2^p} g_n(x_i) x_i^m \Delta x.$$





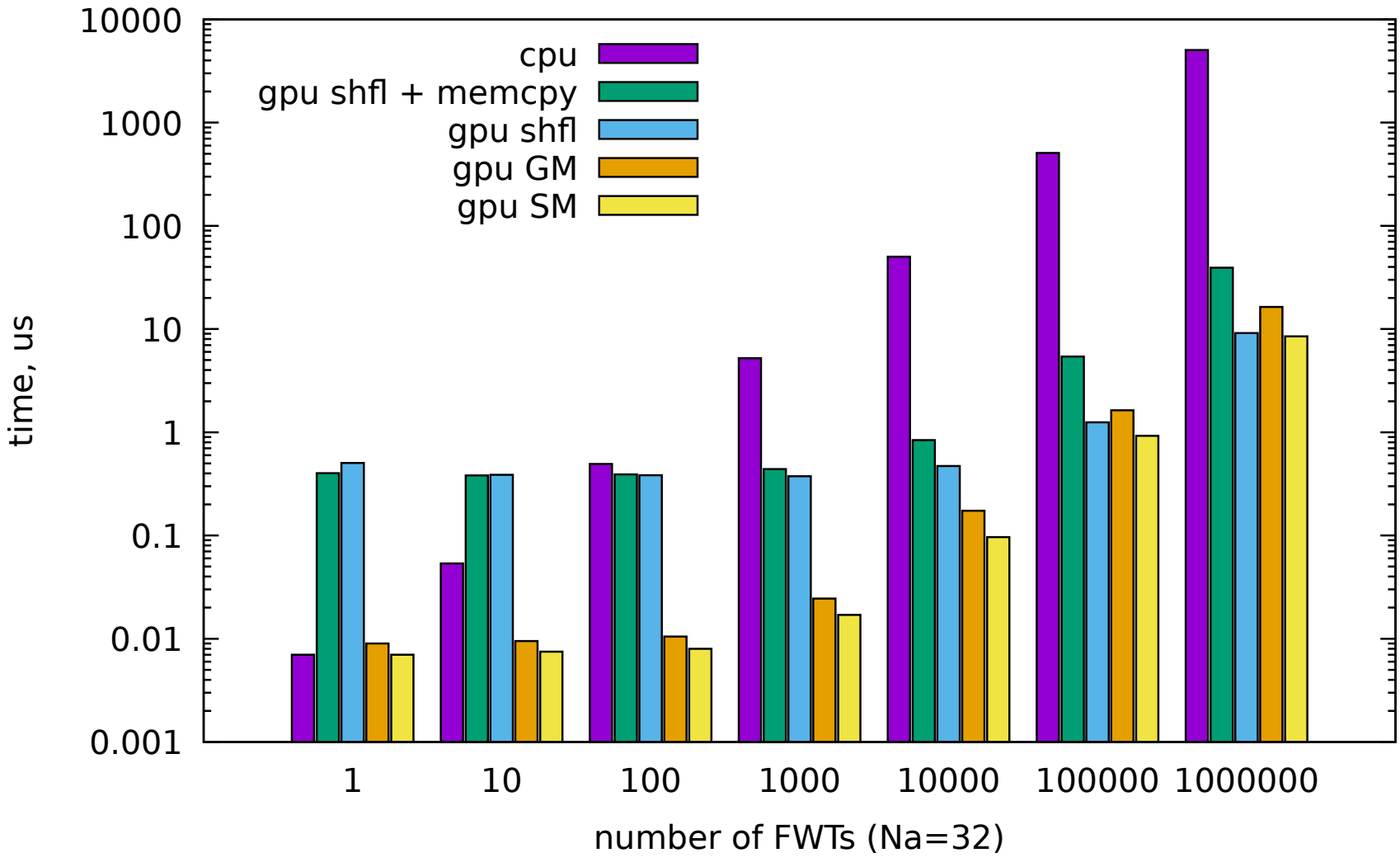
CUDA C Kernel






```
if (tid < N) {
    F1 = fi[(tid / 32) * 32 + seqi];
}

int Nm = Na/2;
for(int pm=0;pm<Pa;pm++) {
    // calculate negMask
    negMask = (((tid >> (Pa-pm-1)) & 1LU) * 2 - 1) * -1; // 1 or -1

    // calculate src mask
    srcMask = ((tid >> (Pa-pm-1)) & 1LU) ^ 1LU; // 0 or 1

    // apply warp shuffle down
    F2 = srcMask * __shfl_down_sync(0xFFFFFFFF, F1, Nm);

    // flip mask
    srcMask ^= 1LU;

    // apply warp shuffle up
    F2 += srcMask * __shfl_up_sync(0xFFFFFFFF, F1, Nm);

    // add to existing warp value, using negMask
    F1 = F1 * negMask + F2;

    // update shfl width
    Nm >>= 1;
}

// write to global memory
if (tid < N) {
    Fa[tid] = F1;
}
```

Results


10000
1000
100
10
1
0.1
0.01
0.001
time, us
cpu
gpu shfl + memcpy
gpu shfl
gpu GM
gpu SM
1
10
100
1000
10000
100000
1000000
number of FWTs (Na=32)