

Linear Algebra Subroutines for Heterogeneous Environments

Bill Brouwer

Research Computing and Cyberinfrastructure
(RCC), PSU

Outline

- **Introduction**
- **Motivation**
- **Applications**
 - Fractional Quantum Hall Effect
 - Image Processing & Radon
 - Drug Delivery
 - Large Scale MD
- **Implementation**
- **Challenges**
 - Amdahl's law revisited
 - Communication
 - Performance
- **Solution**

Introduction

- Research Computing and Cyberinfrastructure (RCC) at PSU provides high performance computing services :
 - **Hardware**
 - **Software**
 - Proprietary/Open Source
 - **Consultation**
 - Numerical
 - Software development
- PhD's, system admins and programmers work together to provide these services to academics while performing independent research
- Many users are interested in using newly acquired NVIDIA GPU cluster for electronic structure and material science, difficult to leverage resources

Motivation

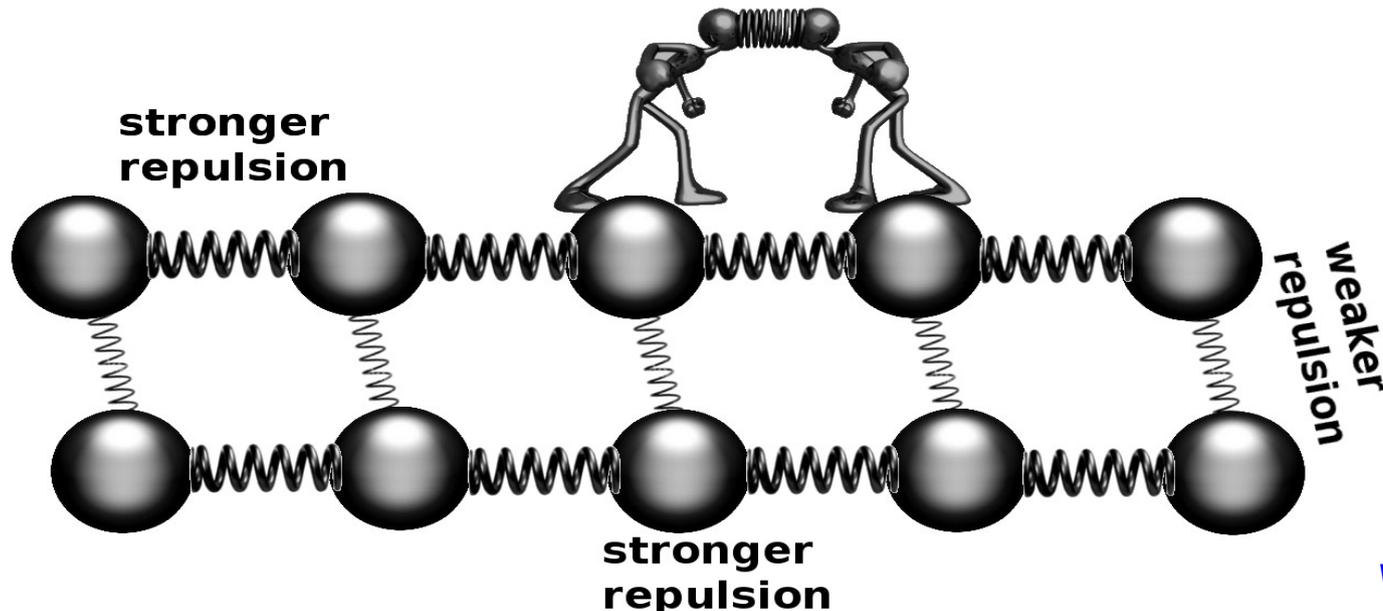
- Scientists like to focus on their research, not necessarily interested and/or experienced in the GPU architecture
- We support and use a variety of excellent GPU enabled resources :
 - **Quantum Espresso (phiGEMM)**
 - **PetaChem**
 - **MAGMA**
 - **LAMMPS**
 - OpenCV
- However many users wish to both scale further & work on unique subsets of (for example) condensed matter physics
- Frequently users are concerned with :
 - Many smaller matrices (eg., FQHE)
 - Several large matrices (eg., DFT)

Outline

- **Introduction**
- **Motivation**
- **Applications**
 - Fractional Quantum Hall Effect
 - Image Processing & Radon
 - Drug Delivery
 - Large Scale MD
- **Implementation**
- **Challenges**
 - Amdahl's law revisited
 - Communication
 - Performance
- **Solution**

Applications : FQHE of $\nu=5/2$

- FQHE at filling fraction $\nu=5/2$ is attributed to a 3 particle interaction between electrons; ground state is the Pfaffian
- Believed to support excitations that exhibit non-Abelian statistics, so far not observed in any other known systems; predictions abound, experiments inconclusive
- One way forward is numerical studies of competing variational wave functions that contain intricate correlations
- Following arrangement emerges at low energy: electrons align in two sets of orbitals, electrons within same set repel more.



Applications : FQHE of $\nu=5/2$

- Each layer is a known many body composite fermion function of $N/2$ particles; difficulty is to combine them into a single antisymmetric function of N particles
- Each evaluation of the whole wave function requires about $N! / (N/2)! (N/2)!$ evaluations
- **Assign the evaluation of these smaller $N/2$ particle functions to different GPU threads \rightarrow many small matrices, each thread assigned complete matrix, performs LU to find determinant**

$$\Psi(1, 2, 3, 4, 5, 6) = \left[\begin{matrix} \text{Antisymmetric} \\ \text{Jastrow} \end{matrix} \right] \times \text{Symmetrization} \{ \phi(1, 2, 3) \phi(4, 5, 6) \}$$

$$\begin{aligned} \text{Symmetrization} \{ \phi(1, 2, 3) \phi(4, 5, 6) \} &= \phi(1, 2, 3) \times \phi(4, 5, 6) + \phi(4, 2, 3) \times \phi(1, 5, 6) \\ &\quad \dots + \phi(2, 4, 3) \times \phi(1, 5, 6) + \phi(1, 2, 4) \times \phi(3, 5, 6) \\ &\quad \dots + \phi(5, 2, 3) \times \phi(4, 1, 6) + \phi(1, 5, 3) \times \phi(4, 2, 6) \dots \end{aligned}$$

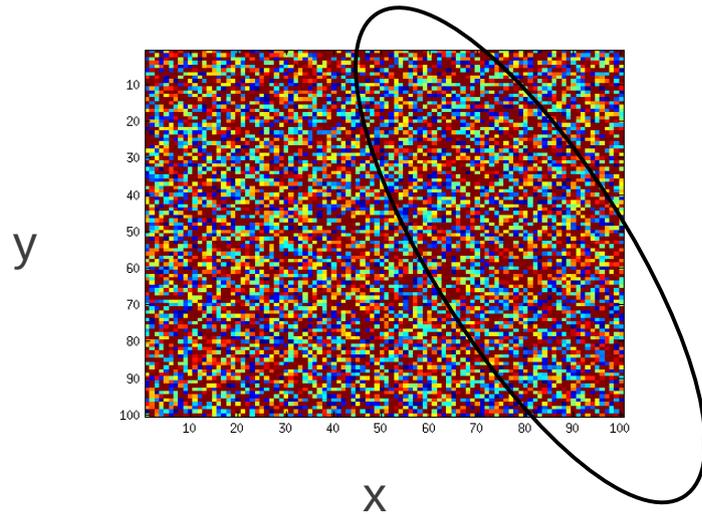
Applications : Radon Transform

- A fundamental, geometric transform relevant in coherent noise removal, spectral editing and other domains eg., computed tomography
- Used in this case to filter figure images, extracted from documents, for incorporation/indexing of figure data in search engine (CiteSeerX)
- (Beylkin) [a image can be viewed as the superposition of different events concentrated along straight lines; RT maps events into points]

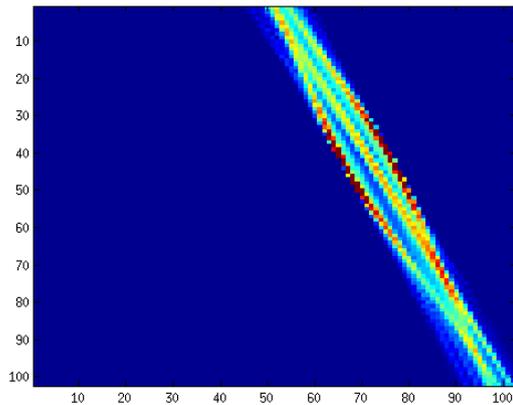
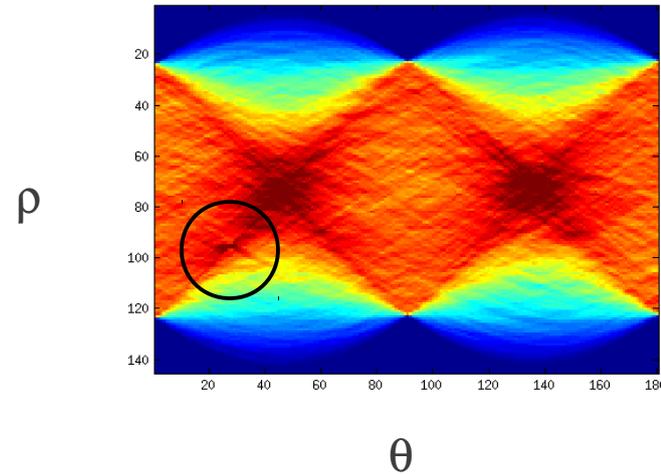
$$d_r(p, \theta) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} d(x, y) \delta[y \sin(\theta) + x \cos(\theta) - p] dx dy$$

Applications : Radon Transform

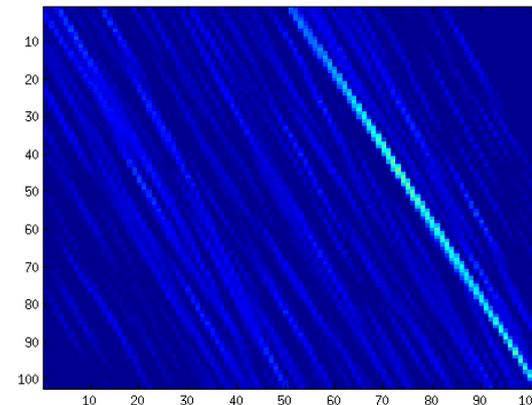
Noisy Image



Radon transform



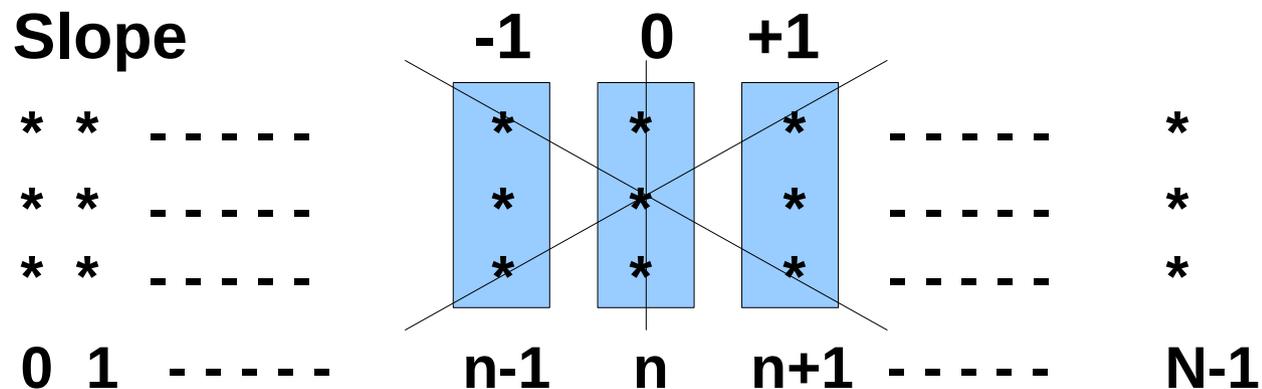
Results of square window & iradon



Results of rectangular window & iradon

Applications : Radon Transform

- (Beylkin) algorithms for numerical evaluation of transform & inverse, based on linear algebra and FFT; consider a simple representation of three sequences & slopes:



- Then a transformed point:

$$y(n) = R_{-1} * x(n-1) + R_0 * x(n) + R_{+1} * x(n+1)$$

- Many small R matrix multiplies (CUDA 4.1/GEMM)

Applications : Lennard Jones Fluid/MD

- Calculation of interactions between all particles through a traditional LJ 6-12 potential, for simulation of combustion properties in propulsion elements :

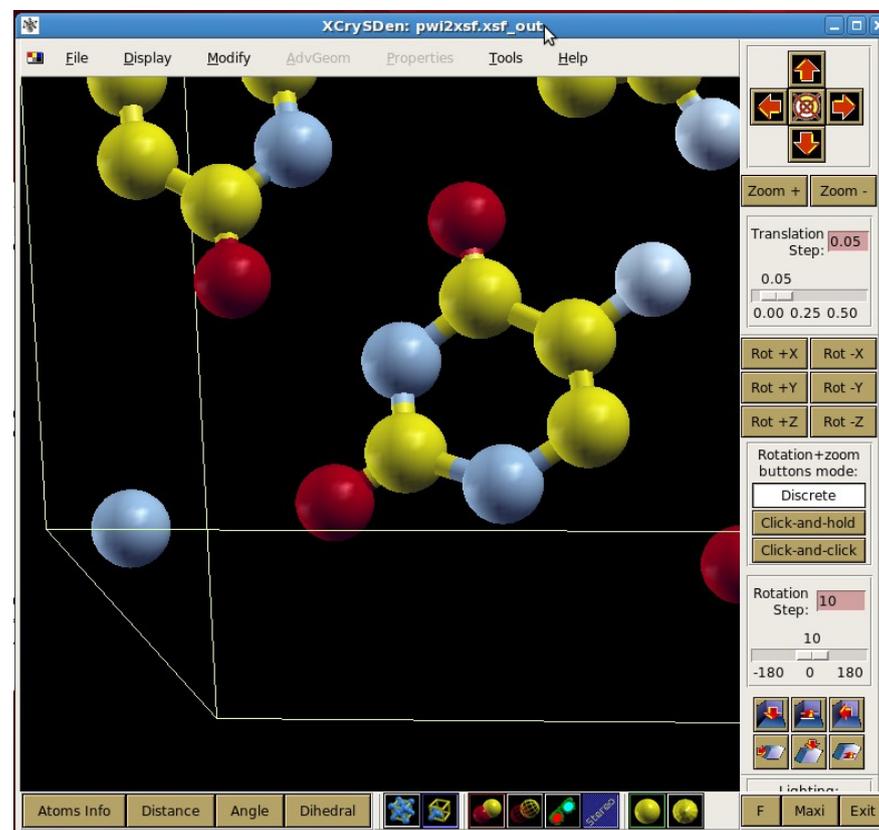
$$V_{ij} = 4\varepsilon \left(\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^6 \right)$$
$$\vec{F}_{ij} = -\vec{\nabla} V_{ij}$$

- Using the Verlet algorithm to find new positions and velocities

$$\left\{ \begin{array}{l} \vec{r}_{n+1} = \vec{r}_n + \vec{v}_n \Delta t + \frac{1}{2} \vec{a}_n \Delta t^2 \\ \vec{a}_{n+1} = \sum \vec{F}_{ij}(\vec{r}_{n+1}) \\ \vec{v}_{n+1} = \vec{v}_n + \frac{1}{2} (\vec{a}_n + \vec{a}_{n+1}) \Delta t \end{array} \right.$$

Applications : Drug Delivery

- Pyrimidine and purine base chemical analogs have long been used in cancer treatment, interfering with DNA synthesis
- Much work has been devoted to understanding exact mechanism of interference, but also delivery methods eg., surface attachment to biomaterials
- The latter can be highly disordered/glassy like; requires very large numbers of atoms in DFT or CP
- Large Matrix evaluations/algorithms needed, particularly diagonalization***

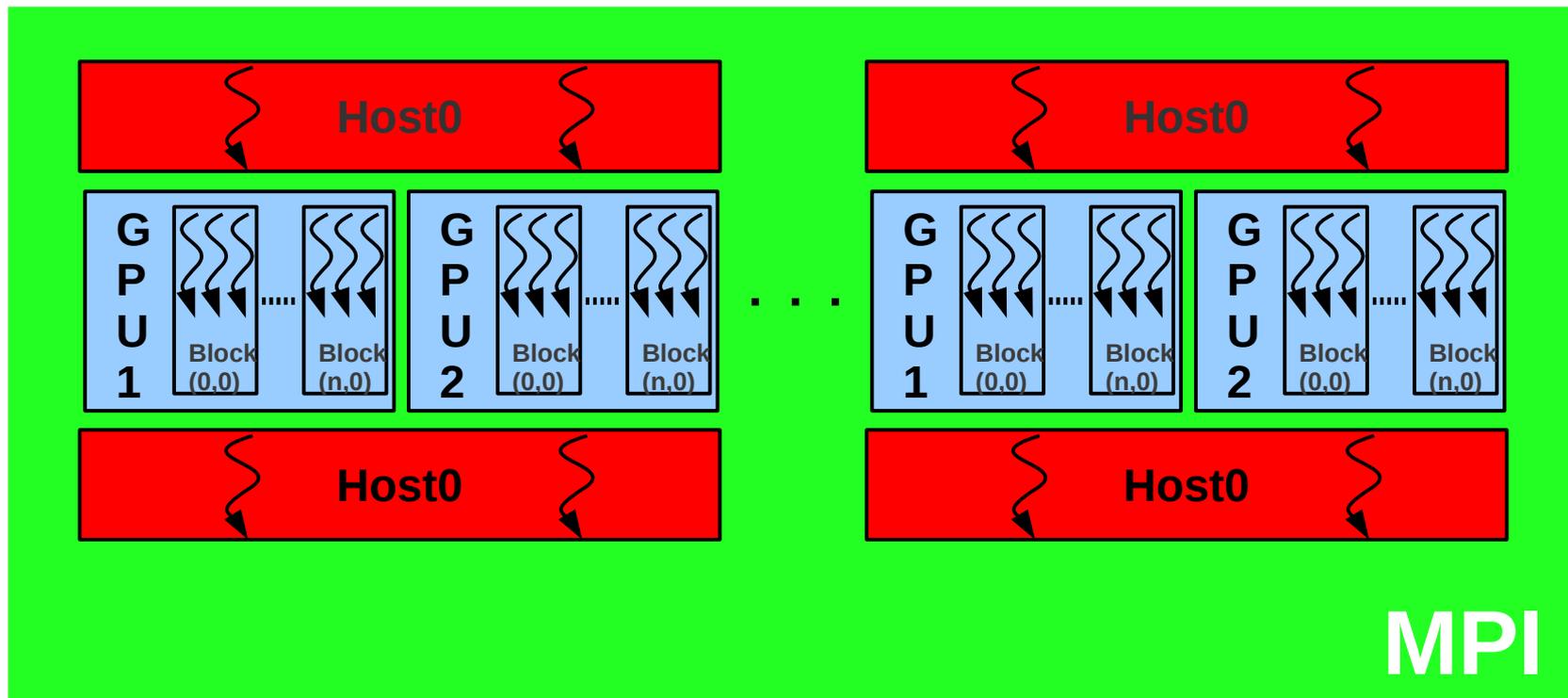


Outline

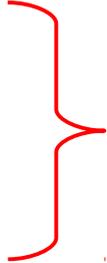
- **Introduction**
- **Motivation**
- **Applications**
 - Fractional Quantum Hall Effect
 - Image Processing & Radon
 - Drug Delivery
 - Large Scale MD
- **Implementation**
- **Challenges**
 - Amdahl's law revisited
 - Communication
 - Performance
- **Solution**

Distributed Heterogeneous Hardware

- Given the performance and scaling required by these applications, ideally we seek to use all ~ three levels of parallelism across clusters at PSU



Compute Elements

- FERMI + CUDA 4.1
 - Unified Virtual Addressing
 - GPUDirect
 - Pinned Memory

allows third party driver(s) to share pinned memory
 - Configurable L1 Cache/Shared Memory
 - NVVP profiler
 - LLVM compiler
- Mellanox IB + OpenMPI
 - GPUdirect support
 - x86 + OpenMP/pThreads/SSE

Outline

- **Introduction**
- **Motivation**
- **Applications**
 - Fractional Quantum Hall Effect
 - Image Processing & Radon
 - Drug Delivery
 - Large Scale MD
- **Implementation**
- **Challenges**
 - Amdahl's law revisited
 - Communication
 - Performance
- **Solution**

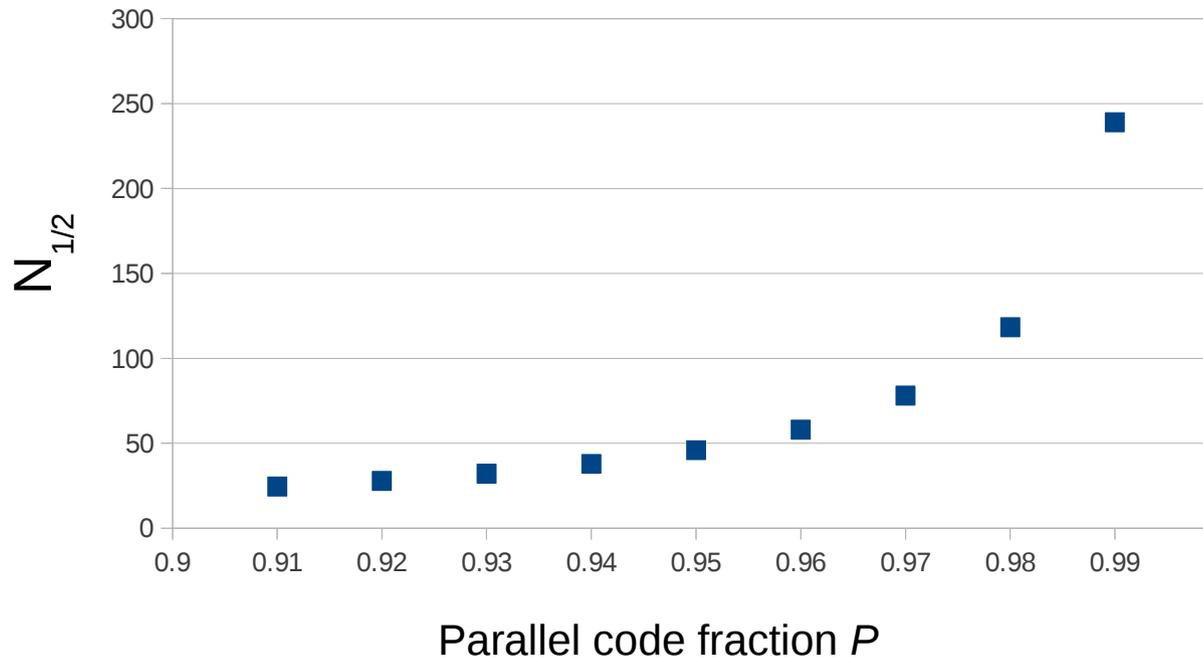
Amdahl's Law revisited

•No less relevant for GPU; in the limit as processors $N \rightarrow \infty$ we find the maximum performance improvement :

$$1/(1-P)$$

•It is helpful to see the 3dB points for this limit ie., the number of processing elements $N_{1/2}$ required to achieve $(1/\sqrt{2}) \cdot \text{max} = 1/(\sqrt{2} \cdot (1-P))$; equating with Amdahl's law & after some algebra :

$$N_{1/2} = 1/((1-P) \cdot (\sqrt{2}-1))$$



Amdahl's law : implications

- Points to note from the graph :

- $P \sim 0.90$, we can benefit from ~ 20 elements
- $P \sim 0.99$, we can benefit from ~ 256 elements
- $P \rightarrow 1$, we approach the “embarrassingly parallel” limit
- $P \sim 1$, performance improvement directly proportional to elements
- $P \sim 1$ implies *independent* or batch processes

- *Want at least $P \sim 0.9$ to justify work of porting code to accelerator*

- For many problems in sciences, as characteristic size increases, P approaches 1 and scaling benefits outweigh cost of porting code (cf weak vs strong scaling)

- *Regardless, doesn't remove the need to **profile** and clearly target suitable sections of serial code*

Profiling w/ Valgrind

```
[wjb19@lionxf scratch]$ valgrind --tool=callgrind ./psktm.x  
[wjb19@lionxf scratch]$ callgrind_annotate --inclusive=yes callgrind.out.3853
```

```
-----  
Profile data file 'callgrind.out.3853' (creator: callgrind-3.5.0)  
-----
```

```
I1 cache:  
D1 cache:  
L2 cache:  
Timerange: Basic block 0 - 2628034011  
Trigger: Program termination  
Profiled target: ./psktm.x (PID 3853, part 1)
```

Parallelizable worker
function is 99.5% of
total instructions
executed

```
-----  
20,043,133,545 PROGRAM TOTALS  
-----
```

Ir	file:function
20,043,133,545	???:0x0000003128400a70 [/lib64/ld-2.5.so]
20,042,523,959	???:0x0000000000401330 [/gpfs/scratch/wjb19/psktm.x]
20,042,522,144	???:(below main) [/lib64/libc-2.5.so]
20,042,473,687	/gpfs/scratch/wjb19/demoA.c:main
20,042,473,687	demoA.c:main [/gpfs/scratch/wjb19/psktm.x]
19,934,044,644	psktmCPU.c:ktmMigrationCPU [/gpfs/scratch/wjb19/psktm.x]
19,934,044,644	/gpfs/scratch/wjb19/psktmCPU.c:ktmMigrationCPU
6,359,083,826	???:sqrtf [/gpfs/scratch/wjb19/psktm.x]
4,402,442,574	???:sqrtf.L [/gpfs/scratch/wjb19/psktm.x]
104,966,265	demoA.c:fileSizeFourBytes [/gpfs/scratch/wjb19/psktm.x]

If we wish to scale outside a single node, we must use some form of *interprocess communication eg., MPI*

Outline

- **Introduction**
- **Motivation**
- **Applications**
 - Fractional Quantum Hall
 - Image processing & Radon
 - Drug Delivery
 - Large Scale MD
- **Implementation**
- **Challenges**
 - Amdahl's law revisited
 - **Communication**
 - **Performance**
- **Solution**

Communication : MPI Grid

- MPI allows one to associate different addressing schemes to processes within a group, necessary for anything but modest tiling schemes
- Either a graph structure or a (Cartesian) grid; need to express:
 - Dimensions {size,period}
 - Option to have processes `reordered` optimally within grid
- Method to establish Cartesian grid `cart_comm` :

```
int MPI_Cart_create(MPI_Comm old_comm, int number_of_dims,  
int dim_sizes[], int wrap_around[], int reorder, MPI_Comm*  
cart_comm)
```

- Remains to be seen how this optimal reordering works with GPUdirect, however bandwidths for simple transactions look promising...

Communication : Infiniband/MPI

- Initial results encouraging for performing network transfers on simple payloads:

```
[wjb19@lionga scratch]$ mpicc -I/usr/global/cuda/4.1/cuda/include  
-L/usr/global/cuda/4.1/cuda/lib64 mpi_pinned.c -lcudart
```

```
[wjb19@lionga scratch]$ qsub test_mpi.sh  
2134.lionga.rcc.psu.edu
```

```
[wjb19@lionga scratch]$ more test_mpi.sh.o2134  
Process 3 is on lionga7.hpc.rcc.psu.edu  
Process 0 is on lionga8.hpc.rcc.psu.edu  
Process 1 is on lionga8.hpc.rcc.psu.edu  
Process 2 is on lionga8.hpc.rcc.psu.edu  
Host->device bandwidth for process 3: 2369.949046 MB/sec  
Host->device bandwidth for process 1: 1847.745750 MB/sec  
Host->device bandwidth for process 2: 1688.932426 MB/sec  
Host->device bandwidth for process 0: 1613.475749 MB/sec  
MPI send/rcv bandwidth: 4866.416857 MB/sec
```

- Single IOH supports 36 PCIe lanes, NVIDIA recommend 16 per device; however IOH doesn't currently support P2P between GPU devices on different chipsets

Perf : Optimization

- Powerful profilers available, now providing excellent hints for improvements
- Performance and scaling still tied to knowledge of architecture

The image displays two software interfaces used for performance optimization. The top interface is the Compute Visual Profiler (nvvp), which provides a detailed summary table of GPU activities and a visual timeline of the application's execution.

nvvp Summary Table:

Method	#Calls	GPU time (us)	%GPU time	glob mem read
1 NBL_GPU	289	3.7513e+06	76.2	0.00579534
2 constr_CELL_cnt	289	361518	7.34	0.000204654
3 kernel_comp	2882	275706	5.6	0.622392
4 zero_NBL	289	32079.3	0.65	0.00462057
5 sum_gpu	6914	31597.6	0.64	0.0119676
6 kernel_acc_int	1	30064.6	0.61	0.0981705
7 gpu_vel_scale	2881	11671.8	0.23	0.0537634
8 fused_sum_gpu	576	10956.8	0.22	1.21822
9 zero	5185	8285.33	0.16	0.00924815
10 kernel_potential	288	7292.38	0.14	6.8587
11 binGPU	289	1432.67	0.02	2.04912
12 zero_CELL_cnt	289	995.488	0.02	0.149514
13 memcpyHtoD	9546	8680.78	0.17	
14 memcpyDtoH	5185	7249.8	0.14	

nvvp Progress Information:

Process: 2242
Thread: 329780160
Runtime API
Driver API
[0] Tesla M2090
Context 1 (CUDA)
MemCpy (HtoD)
MemCpy (DtoH)
Compute
64.1% [300] NBL_GPU (int*, int*, ...)

computeprof Output:

```
firing kernel kernel_comp
kernel result : no error
firing kernel kernel_comp
kernel result : no error
firing kernel NBL_GPU
kernel result : no error
```

nvvp Properties:

Name	Value
Duration	
Session	20.63 s
Timeline	20.526 s
Kernel	13.233 s
Utilization	64.1%
Invocations	300

computeprof

Perf : Precision

- Obvious performance differences btwn single and double precision, much easier to generate fpe in the former
- Solution → mixed precision, algorithms for correction & error checking, removing need for oft-times painful debug

```
#ifndef _FPE_DEBUG_INFO_
#define _FPE_DEBUG_INFO_ printf("fpe detected in %s around %d,
dump follows :\n", __FILE__, __LINE__);
#endif

#define fpe(x) (isnan(x) || isinf(x))

#ifdef _VERBOSE_DEBUG_
    if (fpe(FTx)) {
        _FPE_DEBUG_INFO_
        printf("FTx : %f threadIdx %i\n", FTx, threadIdx.x);
        asm("trap;");
    }
#endif
```

Perf : Driver & Kernel

- Driver is (for most of us) a black box, signals can elicit undesirous/non-performant responses :

```
[wjb19@lionga1 src]$ strace -p 16576
Process 16576 attached - interrupt to quit
wait4(-1, 0x7fff1bfbed6c, 0, NULL)      = ? ERESTARTSYS (To be
restarted)
--- SIGHUP (Hangup) @ 0 (0) ---
rt_sigaction(SIGHUP, {0x4fac80, [HUP], SA_RESTORER|SA_RESTART,
0x3d4d232980}, {0x4fac80, [HUP], SA_RESTORER|SA_RESTART,
0x3d4d232980}, 8) = 0
```

- Driver/kernel bugs, conflicts*

```
[wjb19@tesla1 bin]$ more /proc/driver/nvidia/version
NVRM version: NVIDIA UNIX x86_64 Kernel Module  285.05.09  Fri Sep
23 17:31:57 PDT 2011
GCC version:  gcc version 4.1.2 20080704 (Red Hat 4.1.2-51)
[wjb19@tesla1 bin]$ uname -r
2.6.18-274.7.1.el5          *largely due to user install error :)
```

Perf : Compiler

- Compiler options (indeed compiler choice : `--nvvm --open64`) can make huge differences :
 - Fast math (`--use_fast_math`)
 - Study output of ptx optimizing assembler eg., for register usage
`--ptxas-options=-v`

```
ptxas info      : Compiling entry function '_Z7NBL_GPUPiS_S_S_S_' for
'sm_20'
ptxas info      : Function properties for _Z7NBL_GPUPiS_S_S_S_
768 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info      : Function properties for _Z9make_int4iiii
40 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info      : Function properties for __int_as_float
0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info      : Function properties for fabsf
0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info      : Function properties for sqrtf
0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info      : Used 49 registers, 3072+0 bytes smem, 72 bytes cmem[0], 48
bytes cmem[14]
```

Outline

- **Introduction**
- **Motivation**
- **Applications**
 - Fractional Quantum Hall
 - Image processing & Radon
 - Drug Delivery
 - Large Scale MD
- **Implementation**
- **Challenges**
 - Amdahl's law revisited
 - Communication
 - Performance
- **Solution**

Solution/Summary

- Overall a fairly costly process to create a distributed CUDA/GPU application, particularly unappealing for scientists in other domains with different (niche) interests
- NVIDIA refer to the overall porting process as APOD → *assess, parallelize, optimize, deploy*
- To accomplish this cycle, developers working with scientists must balance many factors:
 - Profiling & readying serial code for porting
 - Optimal tiling & communication
 - P2P, Network, Host/device
 - Compiler and other optimizations
 - Memory hierarchy & architecture
 - Application/kernel/driver interactions
 - Precision etc etc

Solution/Summary

- Developers go through these steps countless times, would be nice to distill the knowledge into ***automatically tuned libraries***, helped by the fact that :
- Many researchers in electronic/atomic/molecular computational sciences use linear algebra subroutines, both large and small in nature, seeking both high performance and scaling in traditional clusters with co-processors/accelerators
- Given the difficulty of writing applications for scaling new science from scratch, much more conducive to produce libraries
- *Can we augment and extend these in the spirit of ATLAS to tune automatically, performing numerical experiments to select optimal tiling, communication patterns, mixtures of precision etc etc??*

Acknowledgements

- PSU Collaborators :
 - Lee Giles (CiteSeer)
 - Jason Holmes, Michael Fenn (RCC)
 - Sreejith Jaya (Physics)
 - Jim Adair, Amra Tabakovic (MatScEng)
 - Jorge Sofo (Physics)
 - Jim Kubicki (Geoscience)
 - Pierre-Yves Taunay (Aero)
 - Nandhini Chandramoorthy (CSE)