Molecular Dynamics, GPUs, and EC2

Scott Le Grand, Principal Engineer, AWS

AMBER

- An MD simulation package and a set of MD force fields
- PMEMD is a subset of the MD simulation package optimized for production use
- 12 versions as of 2012
- GPU support since AMBER 11

Molecular Dynamics on GPUs

- On a CPU, the dominant performance spike is:

```
for (i =0; i < N; i++)
    for (j = i + 1; j < N; j++)
        Calculate fij, fji;
```

If we naively ported this to a GPU, it would die the death of a thousand race conditions and memory overwrites

Solution: Map the problem into many subtasks and reduce the results

Two Ways to Map, Many Ways to Reduce

Subtasks can be determined by:

1. Dividing work into spatially separated calculations that dump their results into unique preassigned accumulation buffers for later reduction

2. If memory is tight or the problem is large, then further divide the calculation into discrete phases that recycle a smaller subset of such buffers

CUDA port relies on method #1

3 Precision Models

DPDP	Double-precision forces and accumulation
SPDP	Single-precision forces, double-precision accumulation
SPSP	Single-precision forces, single-precision accumulation

Why Multiple Precision Models

- Double-precision on GPUs eats registers and cache but is potentially overkill applied across the board
- Suspected single-precision was insufficient
- SPDP hopefully hits the sweet point between them

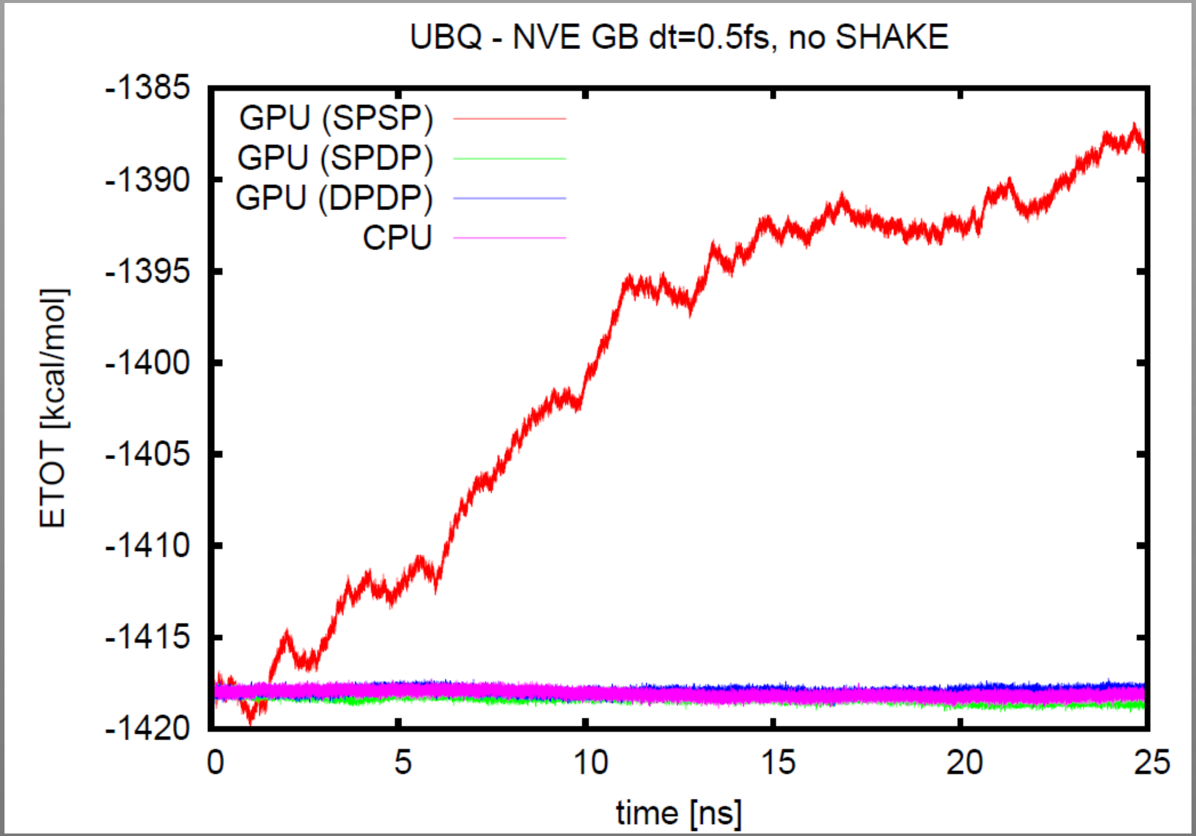
Dynamic Range

32-bit floating point has approximately 7 significant figures

1.456702	1456702.0000000
+0.3046714	+ 0.3046714
---------------	-------------------------
1.761373	1456702.0000000
-1.456702	-1456702.0000000
--------------	-------------------------
0.3046710	0.0000000
Lost a sig fig	Lost everything.

When it happens: PBC, SHAKE, and Force Accumulation.

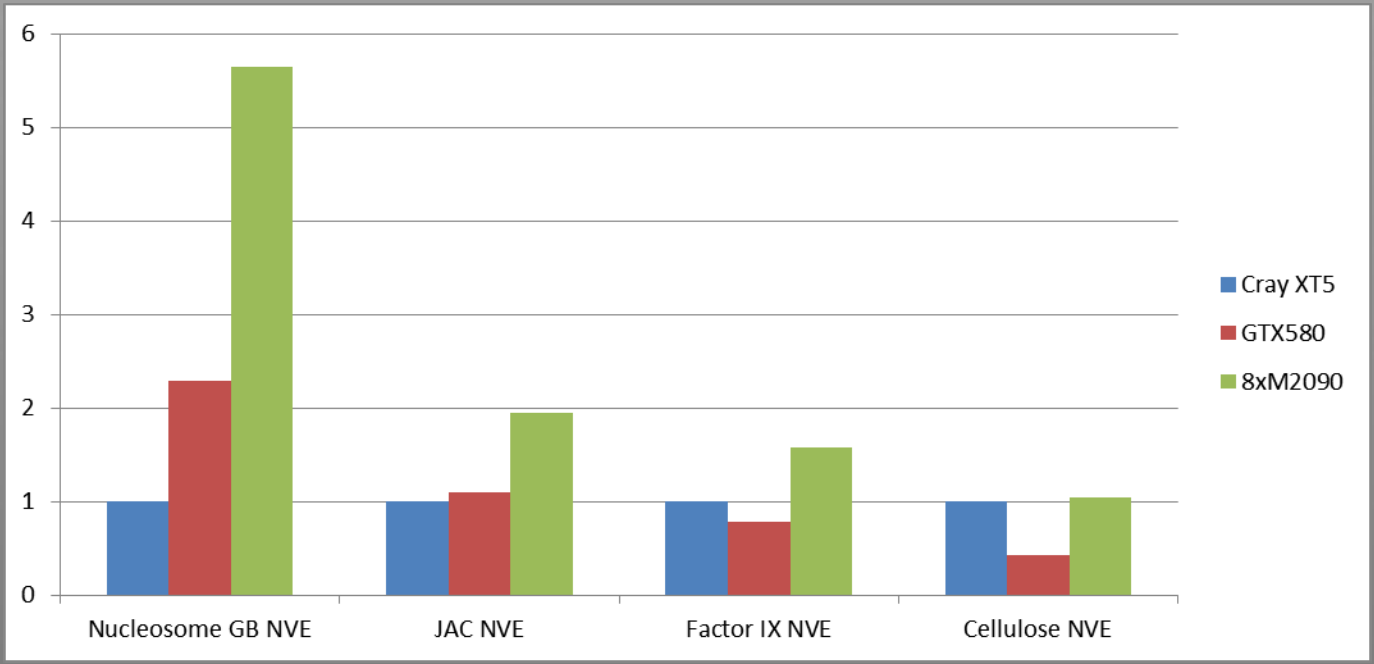
Dynamic Range Matters… A Lot…



Energy Conservation, Explicit Solvent (kT/ns/d.o.f)

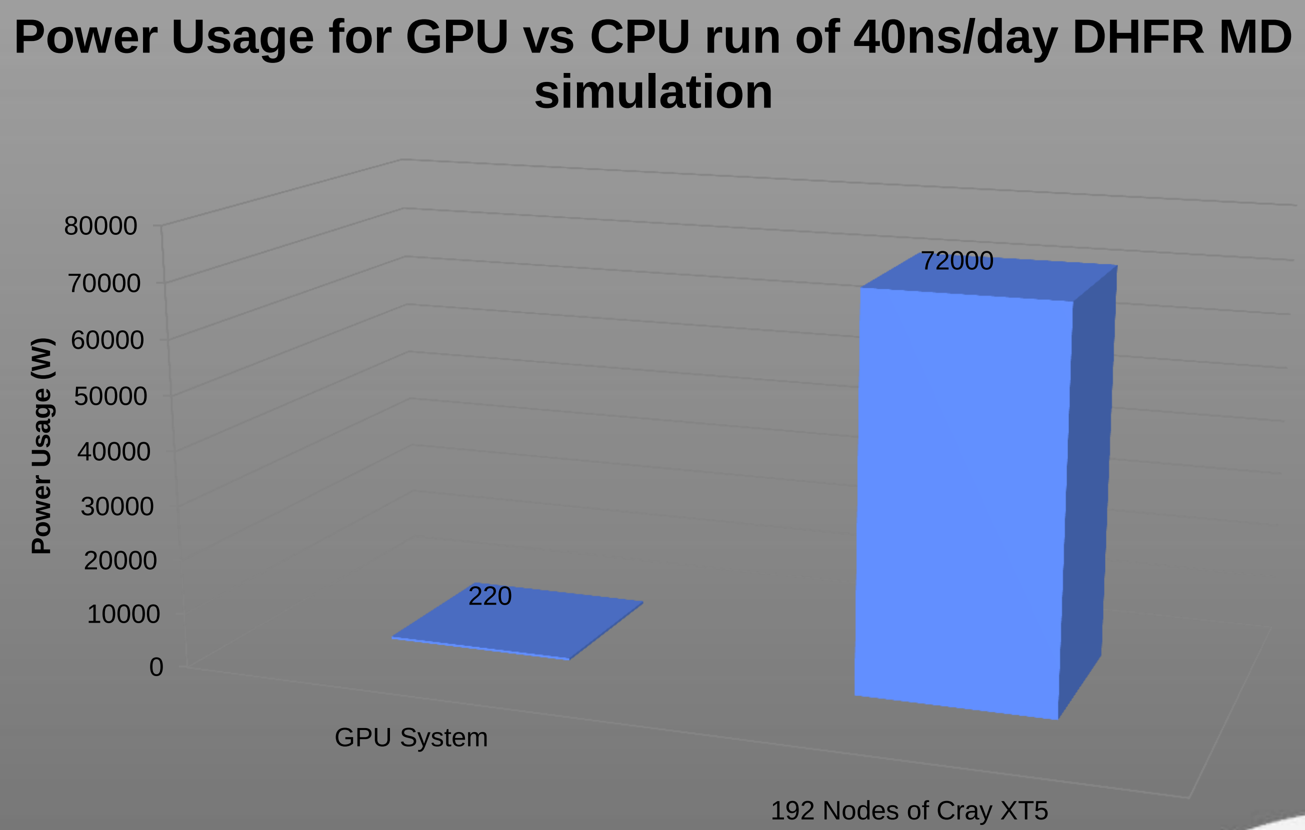
DHFR	dt=1.0fs	dt=2.0fs, SHAKE
CPU	0.000001	-0.000047
DPDP	0.000024	-0.000101
SPDP	0.000050	-0.000066
SPSP	0.001171	3.954495
Gromacs 4	0.011000	0.005000
Desmond	0.017000	0.001000
NAMD	0.023000	----------

(Relative*) Performance



*Relative to 48 to 256 nodes of a Cray XT5 that is…

It's Green™ too…



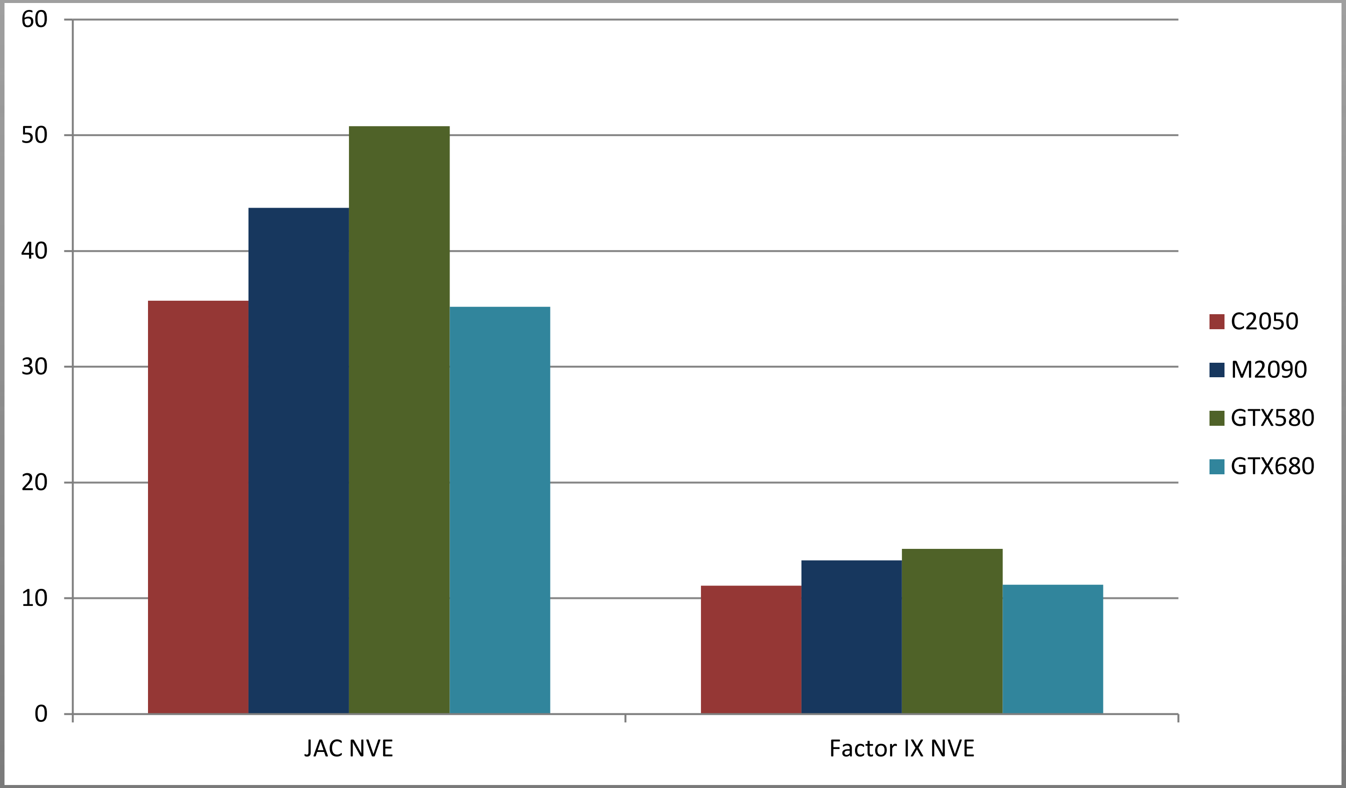
You Don't Know JAC (Production DHFR Benchmark)…

- JAC stands for the Joint AMBER CHARMM Production DHRF Benchmark
- JAC Production DHFR Benchmark specifies a specific combination of specific simulation conditions (especially the timestep)
- Change any one of those conditions and it's no longer the JAC Production DHFR Benchmark (especially the timestep)
- If we're going to allow the equivalent of steroids and blood-doping here, I can probably take AMBER to ~1 microsecond/day on a single GPU but is that productive or useful to anyone but marketing?
- Can't we all just get along?

GTX 680: The Bad News First…

- Naively ported GPU apps suffer miserably due to:
  - Increased operational latency (768 threads per SM increases to 1,280 threads per SMX)
  - Despite this, each Kepler SMX has the same amount of shared memory as a Fermi SM
  - Lousy double-precision performance (~128 GFLOPs)
  - Inefficient allocation of registers by nvcc leads to increased spillage and wasted registers
  - But hey, it's a low-end chip, what do you expect?

Initial GTX 680 Performance



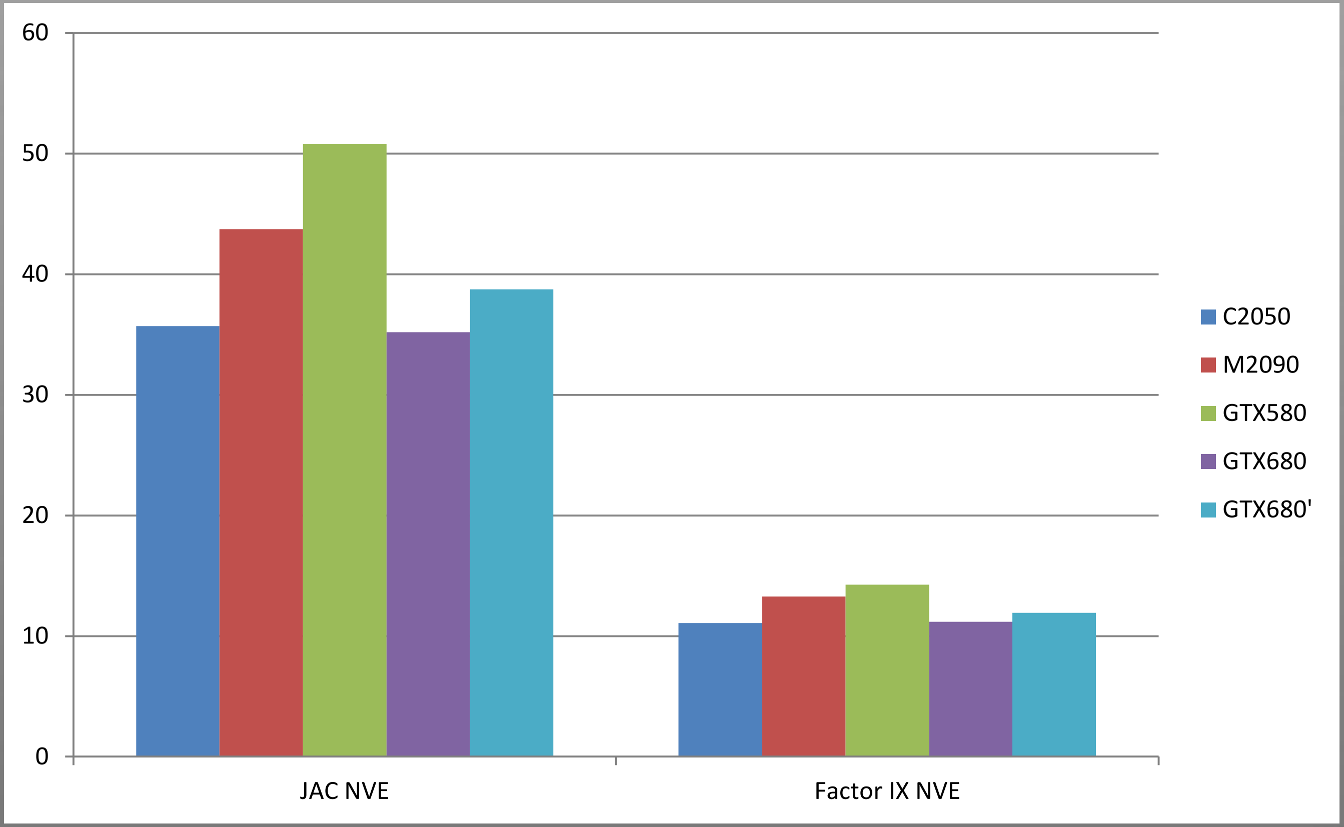
GTX 680: Now The Good News

- Despite my initial shock and grumbling, I *love* GTX 680
  - Twice the registers
  - 3x faster Atomic Ops
  - Twice the single-precision and integer ops
  - __shfl allows warps to share register data as a workaround for not adding more shared memory

Use shfl and registers instead of SMEM

- Move 32-bit quantities shared within warps to registers
- Access them with __shfl instruction
- Now increase the thread count to exploit the freed up SMEM
- Leave 64-bit quantities in SMEM (no 64-bit shfl op)
- Finally, set SMEM to 64-bit mode

Still pretty bad…



Desperation ensues…

- A few years back, Tetsuo Narumi wrote a cool paper about using 6-12 single-precision ops and two variables to approximate 48-bit double-precision:

  Narumi et al., High-Performance Quasi Double-Precision Method using Single-Precision Hardware for Molecular Dynamics Simulations with GPUs, *HPC Asia and APAN 2009 Proceedings*

- Tried it, but it fell short of the claimed precision for me
- But in the middle of evaluating this approach, I tried out 64-bit fixed point…

Use 64-bit fixed point for accumulation

- Each iteration of the nonbond kernel in PMEMD used 9 double-precision operations
- Fermi double-precision was ¼ to 1/10th of single-precision
- Kepler double-precision is 1/24th single precision!
- So accumulate forces in 64-bit fixed point
- Fixed point forces are *perfectly* conserved
- 3 double-precision operations per iteration
- Integer extended math is 32-bit!

Use atomic ops for 64-bit RMW

- Fermi and Kepler have roughly the same GMEM bandwidth
- So the only way to improve on this is to do fire and forget RMW (read-modify-write) operations
- Also eliminates clearing and reducing of accumulation buffers
- Dramatically lower memory footprint
- Still deterministic!

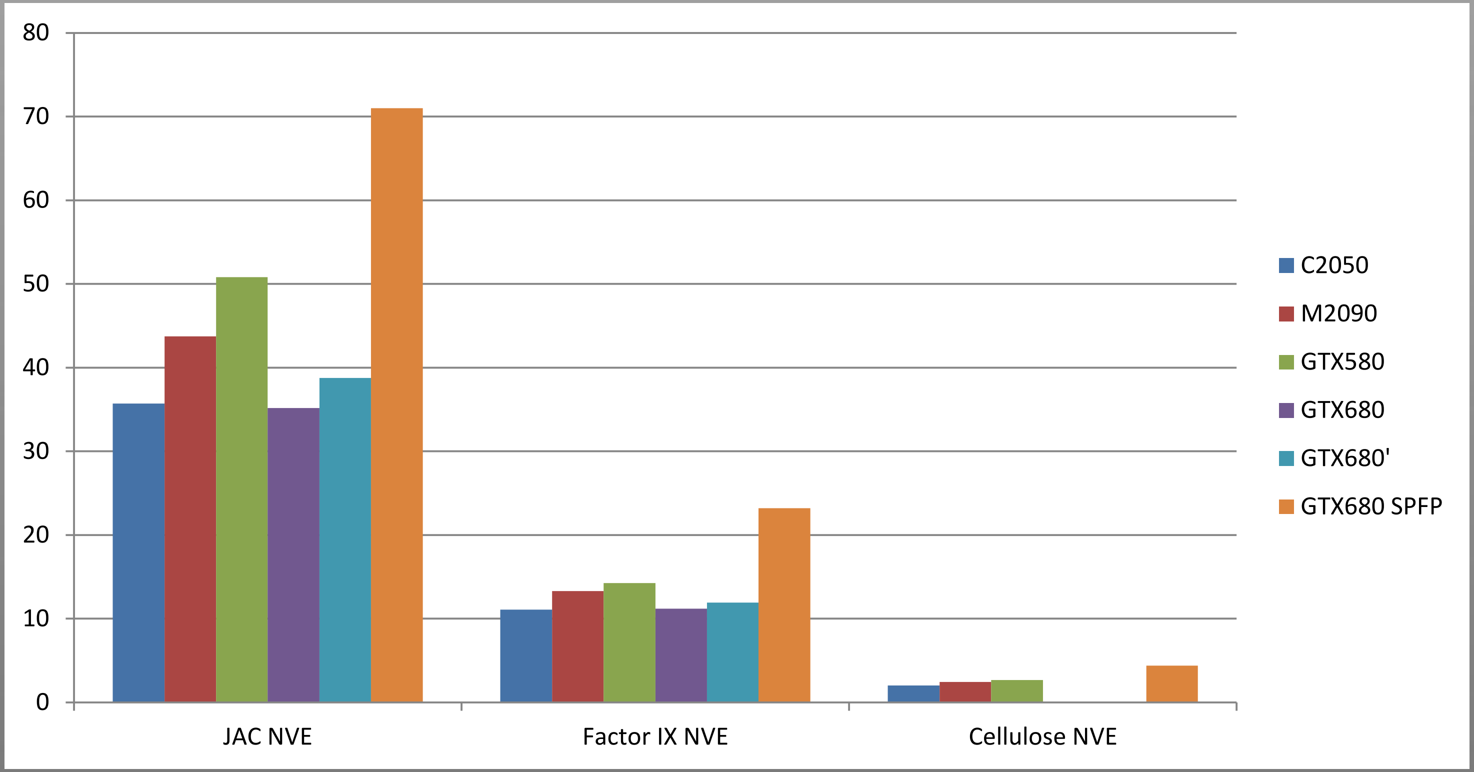
Use atomic ops for Ewald sum

- Ewald sum previously used 8 to 27 floating-point values per charge grid point to prevent race conditions
- Atomic ops eliminate the need for this
- Slower on Fermi, 50% faster on Kepler
- Also still deterministic!

Move irregular SMEM data to L1

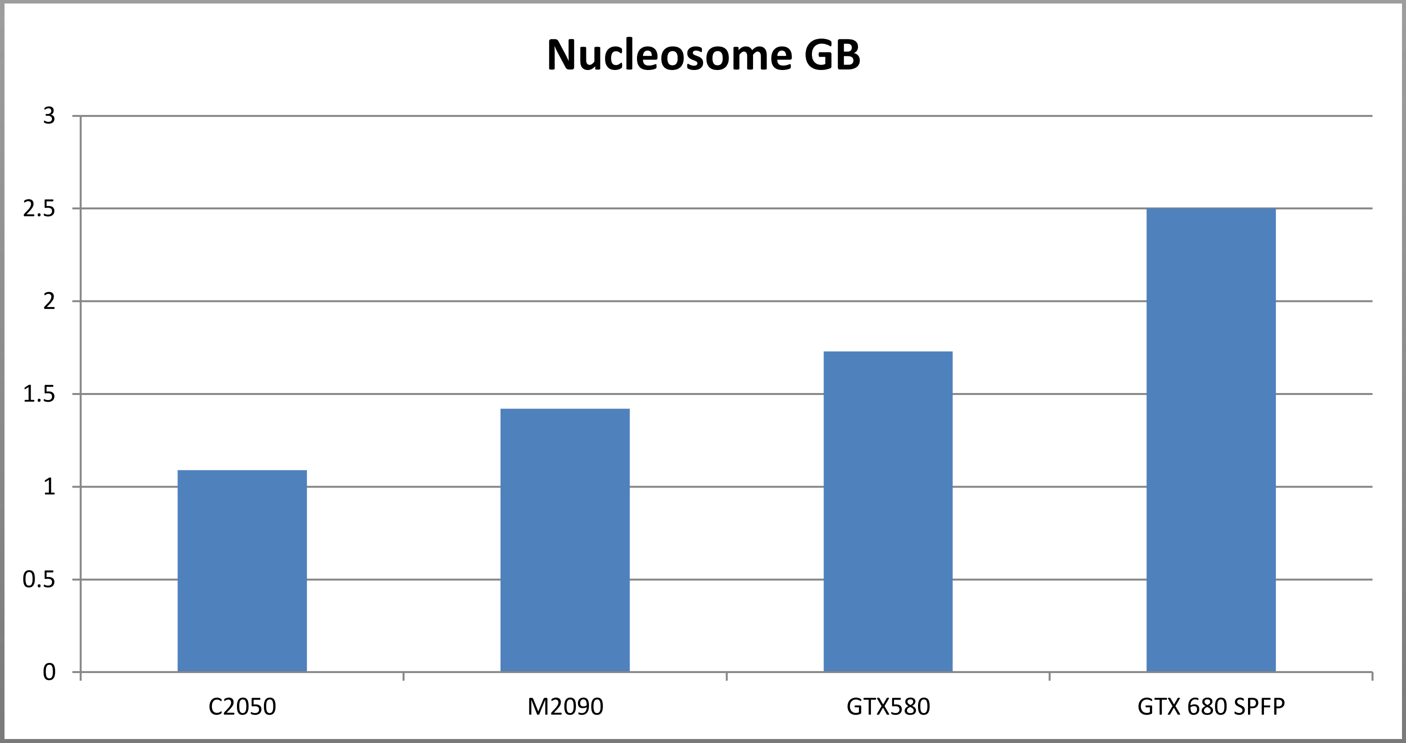
- Neighbor List construction needs a table of bonded atom pairs called “exclusions” that are used for ignoring their potential nonbond interactions
- There was no way to fit this in SMEM and increase thread count to account for the increased operational latency
- Unexplored: In fixed point, one does not need to worry about how these are filtered out – they can be post-procssed

GTX 680 SPFP Performance* (PME)



*~40-50% faster than GTX 580

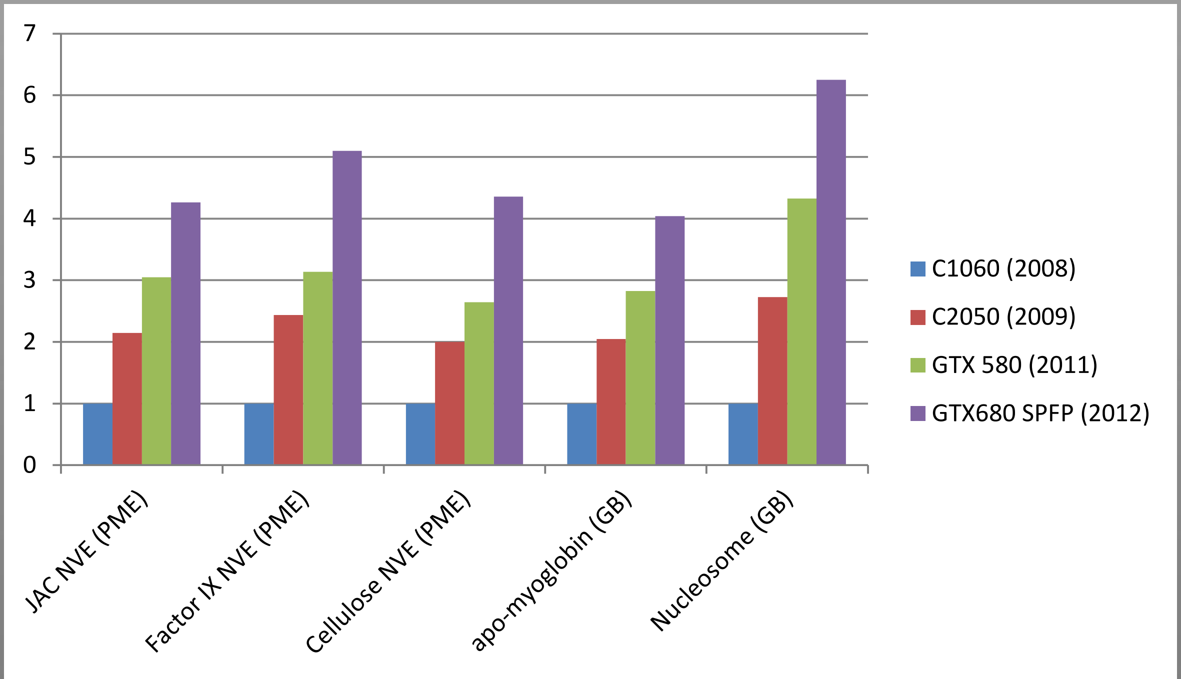
GTX 680 SPFP Performance (GB)



Energy Conservation, Implicit Solvent (kT/ns/d.o.f)

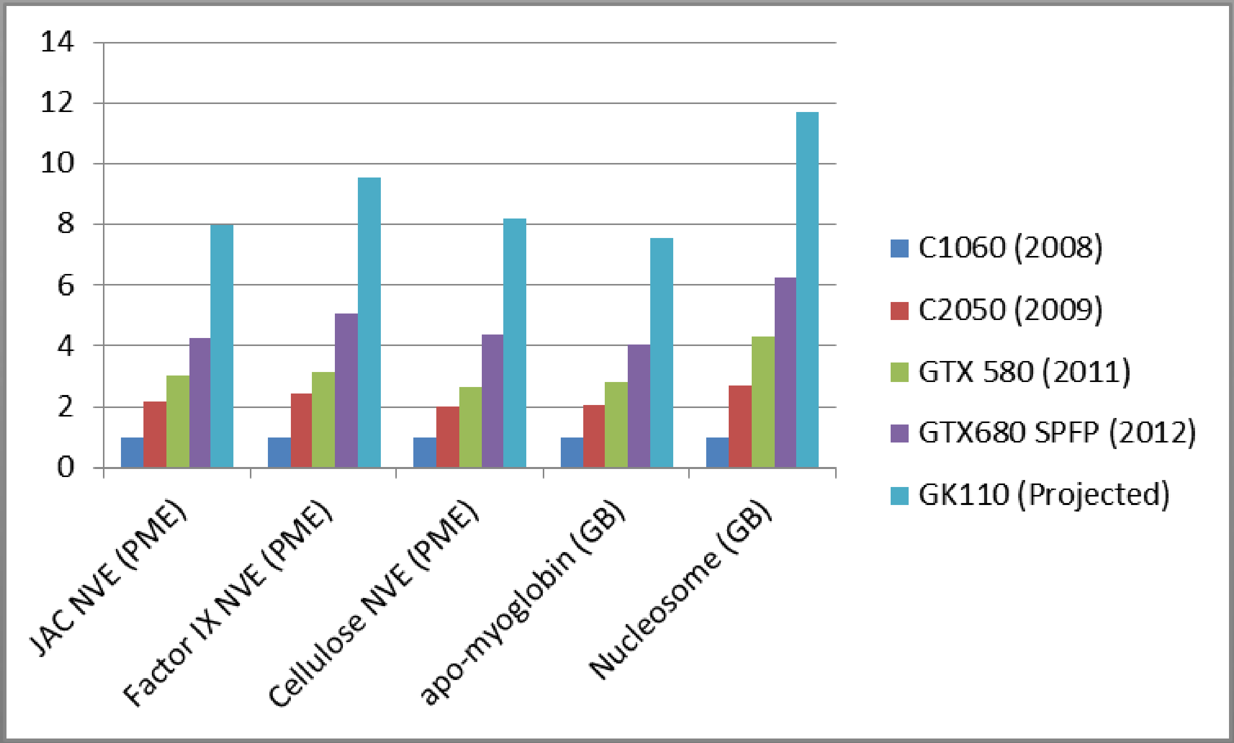
apo-myoglobin GB	dt=1.0fs	dt=2.0fs, SHAKE
CPU	0.000094	0.000416
DPDP	0.000117	0.000290
SPDP	0.000185	0.000139
SPFP	0.000122	0.000254

Summary: 4-6x faster over 3 years



“This isn’t science, this is engineering.” – Anonymous Competitor

GK110: Expect 8-12x by the end of the year*…

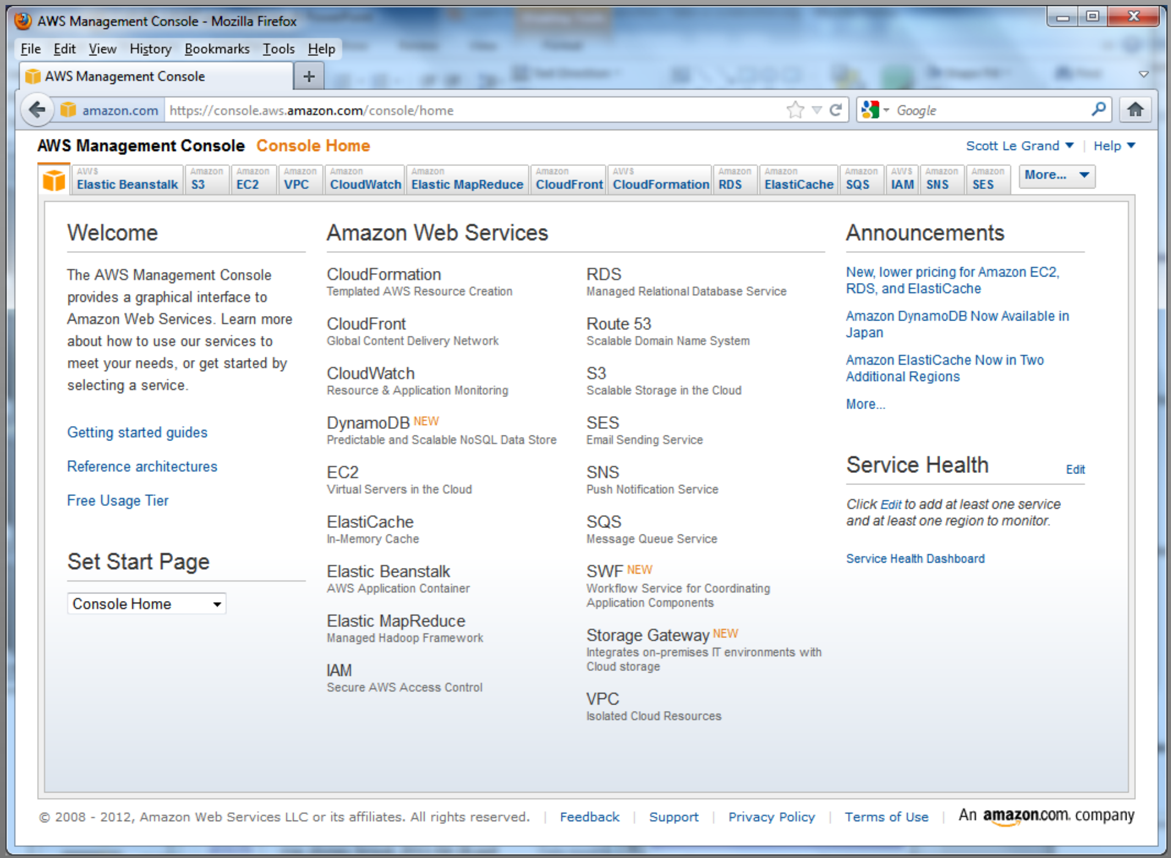


*Based on 15 SMXs, a 384-bit memory bus, and Ge Force Clocks

What The Cloud Isn't…

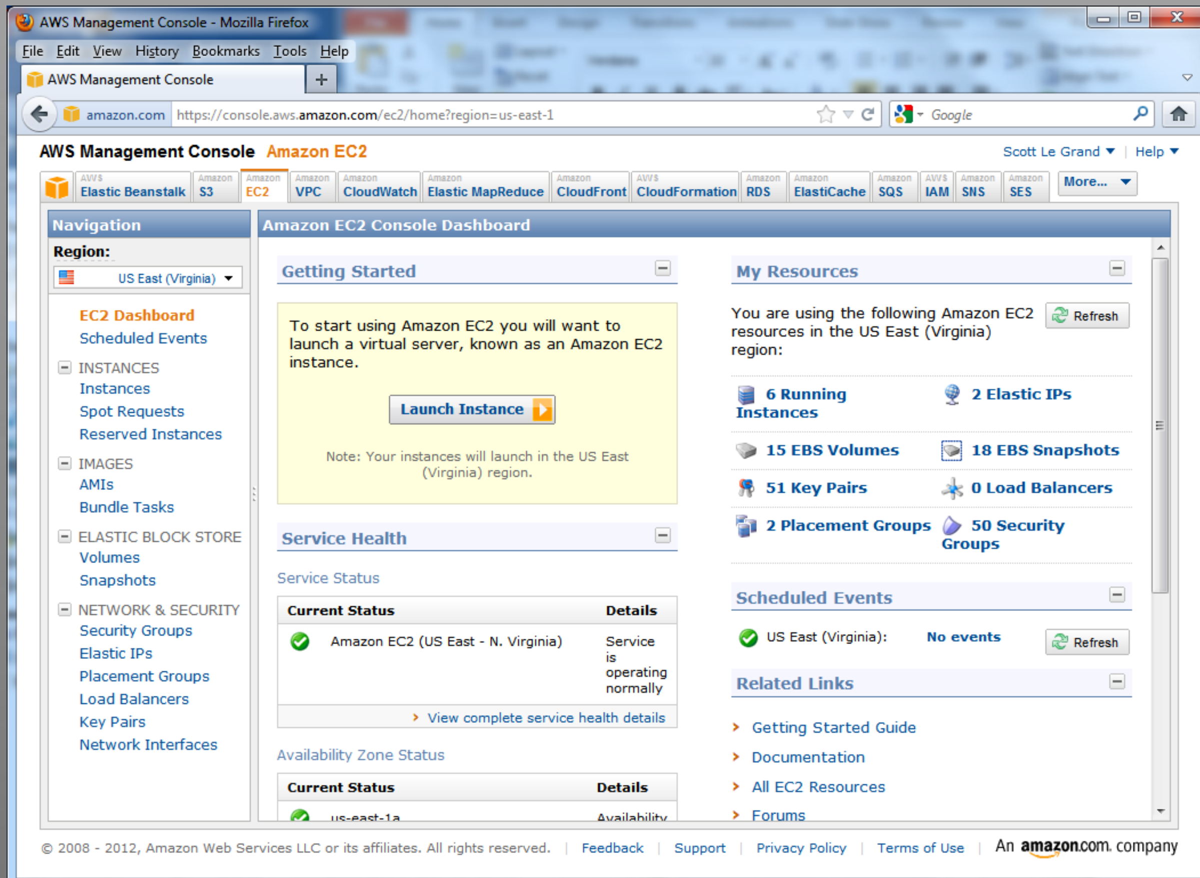
1. Port Application to Cloud
2. ???
3. PROFIT!!!

What AWS (Amazon Web Services) Is…



amazon webservices™

Infrastructure As a Service



All sorts of computer types (instances)

- cc1.4xlarge – 2 quad-core CPUs
- cc2.8xlarge – 2 octa-core CPUs
- cg1.4xlarge – 2 quad-core CPUs + 2 C2050 GPUs

But Why Bother?

Port Once, Run Everywhere (from a web page)

Avoids situations like this:

On Mon, Mar 05, 2012, Yudong Sun wrote:
>
> I have got the following error in compiling AmberTools 1.5 with Amber 11
> using gcc 4.6.1:
>
> (cd nab && make install)
> make[1]: Entering directory
> `/esfs2/z03/z03/ydsun/queries/q202726_amber11/amber11/AmberTools/src/nab'
> ./nab -c dna3.nab
➢ nab2c failed!

Deploy Previously Licensed Software For Fun and Profit (without a license)

Build a single consistent image for your application

Use Identity and Access Management (IAM) to control access

If you’re ambitious, build CLI and web tools to abstract away any notion of the cloud

Monetize with Amazon Marketplace

Get Extra Capacity When You Need It

- Lead optimization is only embarrassingly parallel if you have the hardware to make it that way
- Do you want to wait 3 months or 3 days to finish those free energy calculations?
- The easiest way to reduce sampling error is to do more sampling
- Public cloud is oversubscribed

amazon webservices™

Manage it all with StarCluster

- Python application that allows one to dynamically spawn clusters of any sort and size on EC2
- Comes with Open Grid Engine preinstalled
- Dynamically adjusts cluster size based on size of job submission queue
- Automagically manages GPU usage if your app intelligently handles exclusive mode
- http://serverfault.com/questions/377005/using-cuda-visible-devices-with-sge

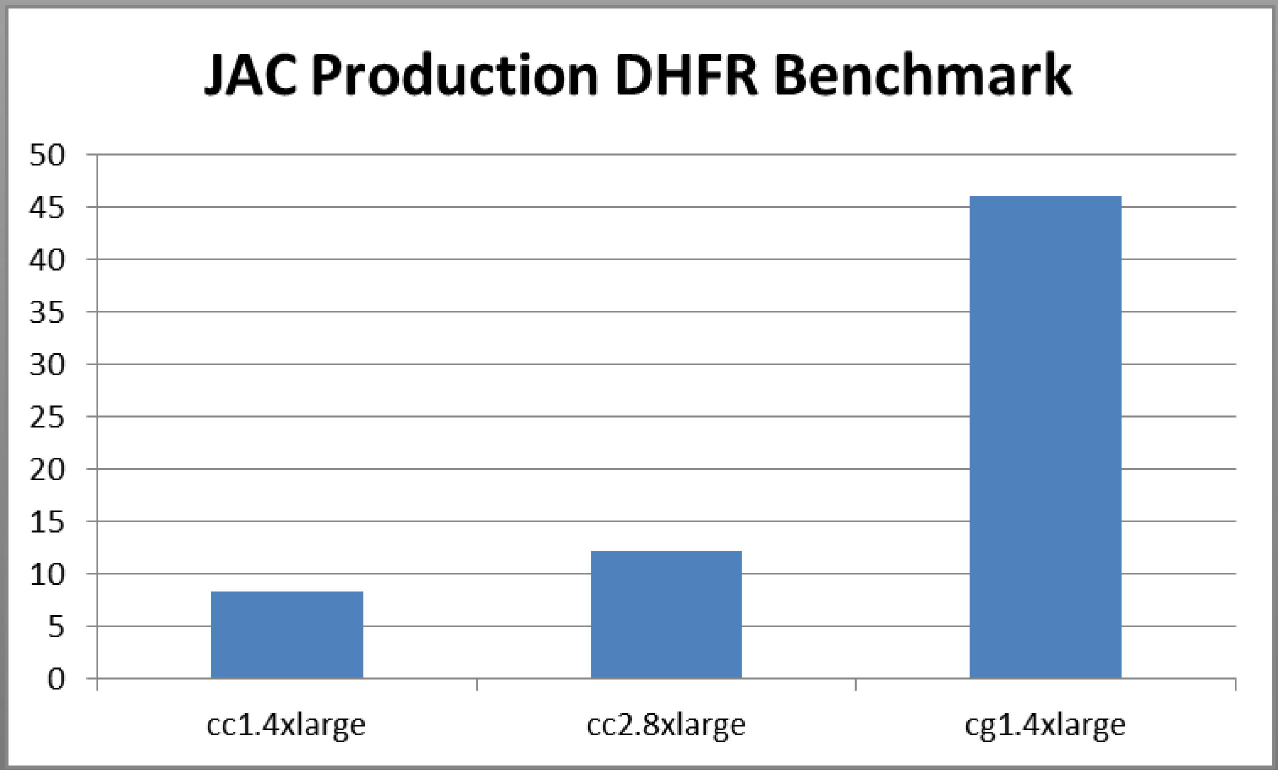
```
// Let CUDA select any device from this list of device IDs filtered by your
// own criteria (not shown)
status                    = cudaSetValidDevices(pGPUList, nGpus);
if (status != cudaSuccess)
{
   printf(("Error searching for compatible GPU\n");
   exit(-1);
}

// Trick driver into creating a context on an available and  valid GPU
status                    = cudaFree(0);
if (status != cudaSuccess)
{
   printf("Error selecting compatible GPU\n");
   exit(-1);
}

// Get device selected by driver
status                    = cudaGetDevice(&device);
if (status != cudaSuccess)
{
   printf("Error fetching current GPU\n");
   exit(-1);
}

// Your amazing CUDA program goes here...
```

Which instance type is the fastest?



But don’t take my word for it…

- NVIDIA and AWS are sponsoring the GPU Test Drive
- http://www.nvidia.com/gputestdrive
- Run PMEMD anywhere on your own data
- $100 (~48 hours of cg1.4xlarge EC2 time) for free…

Two Examples

- Ensemble Molecular Dynamics
- 100s to 1000s of independent runs with subsequent analysis of results
- We have 100s to 1000s of GPUs standing by
- Get a year’s worth of molecular dynamics in a day
- Store intermediate results in S3

Replica Exchange Molecular Dynamics

- 100s to 1000s of loosely coupled molecular dynamics simulations periodically exchanging data
- Network fabric is the rate limiter
- Runs at 85-90% of peak for 100+ replicas over 50+ instances

What We're Doing…

Everything we can to make porting these applications to the cloud as simple as possible

Talk to us!

Summary

- GPUs bring enormous gains in computational firepower and memory bandwidth that just keep on coming…
- EC2 lets you scale when you need to and provides brand new ways to deploy your applications…
- Kepler: Learn to love it because the best is yet to come…

Acknowledgments

AWS: Deepak Singh, Mike Marr, Matt Wood

NVIDIA: Mark Berger, Duncan Poole, Sarah Tariq

AMBER: Ross Walker, Jason Swails, David Case

And Finally…

I'd like to acknowledge my father, Donald Le Grand

4/2/1930-3/7/2012

Without whom, none of this was possible…