Trends in High Performance Computing, and their Impact on Astrophysical Data Processing

Theodore Kisner
Computational Cosmology Center, LBNL

GRITS — 2010/06/11

C^3 at LBNL

- Focused on computational challenges (simulation and data processing) relevant to cosmology (CMB, SN, BAO, ...)
- Tight connection to DOE computing facilities: Cray XT5 (40K cores), Cray XE6 (150K cores), Cloud computing platform, GPU test cluster, science gateways, etc.
- For >10 years, we have coordinated CPU allocations for CMB telescopes (funded by NASA, NSF, etc).
- Involved in building software infrastructure for future experiments and future architectures: algorithm scaling, data management, etc.

High Performance Computing

For the purposes of this talk, everything that needs a machine room:

- Traditional Clusters (PCs interconnected with ethernet, infiniband, etc)
- Supercomputers (lightweight nodes with infiniband or custom interconnect)
- Cloud computing platforms (EC2, Eucalyptus)
- Large shared memory machines (NUMA architectures)

HPC in 10 Years

Hard to predict, but driven by trends:

- Still using silicon, and still tracking Moore's law for transistor counts.
- Computing centers have limited electrical capacity for power and cooling.
- Packing transistors into traditional CPU cores requires even more transistors for "overhead"-diminishing returns.
- Market forces (follow the money)

Moore's Law

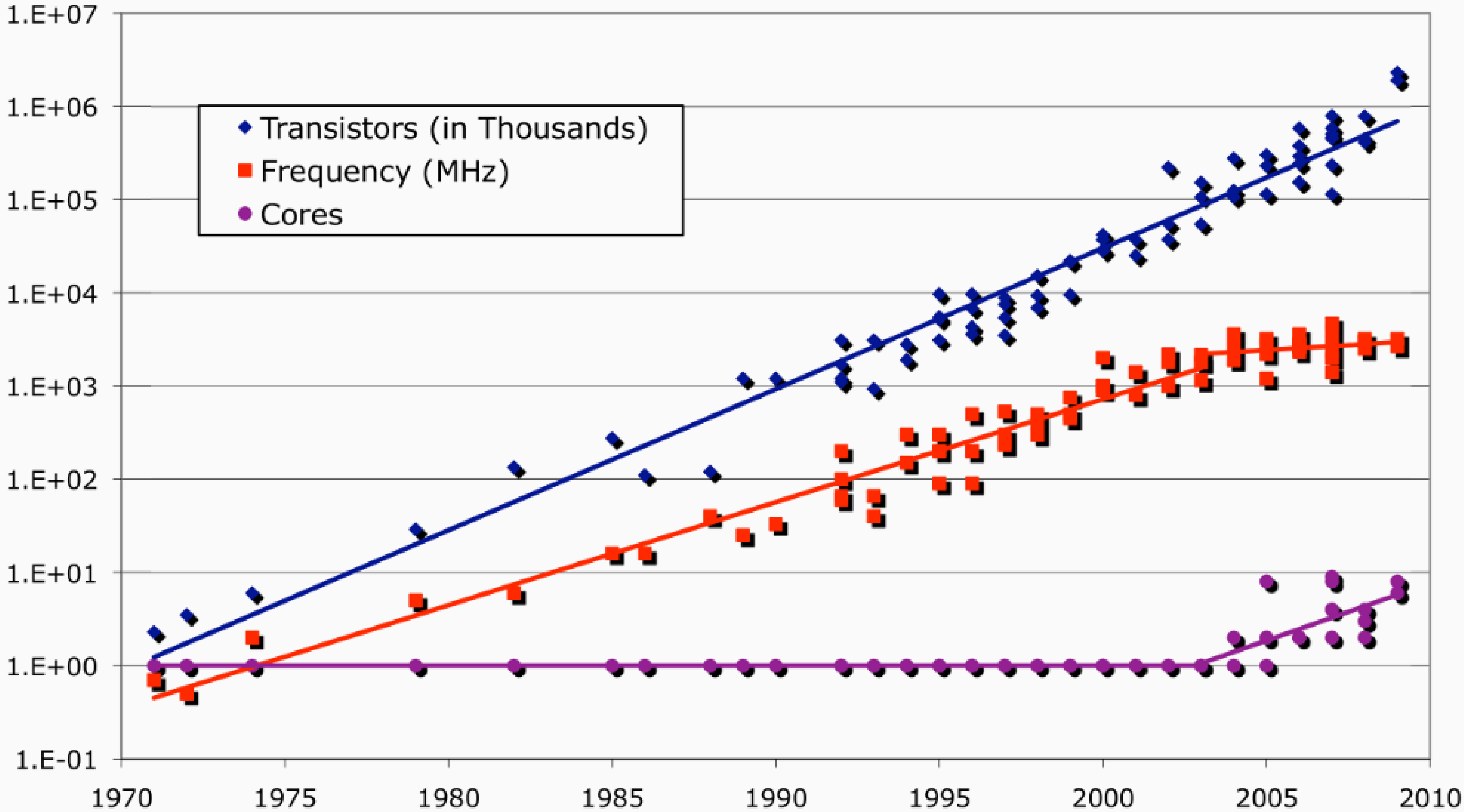


Figure by Kathy Yelick, data from Kunle Olukotun, Lance Hammond, Herb Sutter, Burton Smith, Chris Batten, and Krste Asanovic

Rise of Many-core Systems

Focus is on Flops per Watt:

- Clock rates constant or decreasing.

Clock Rates and Power Scaling

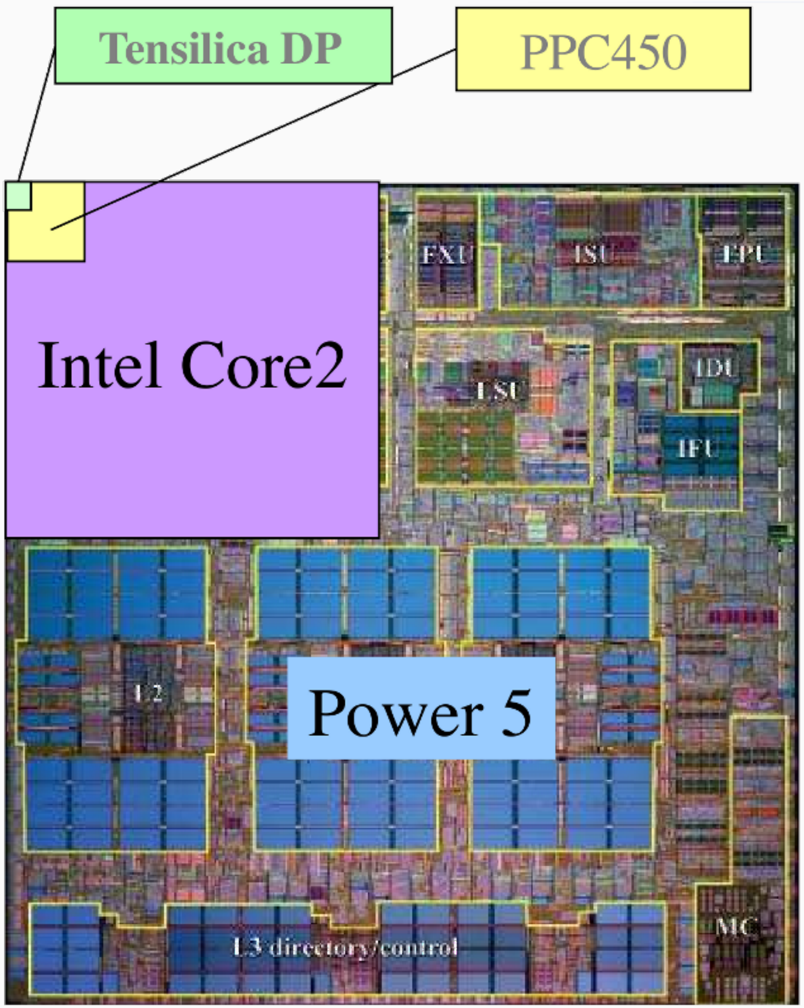


Image by John Shalf, LBNL

- IBM Power5: 120W @ 1900MHz
- Intel Core2 solo: 15W @ 1000MHz.
- IBM PPC 450 (Blue Gene): 0.625W @ 800MHz
- Tensilica XTensa (Moto Razor): 0.09W @ 600MHz

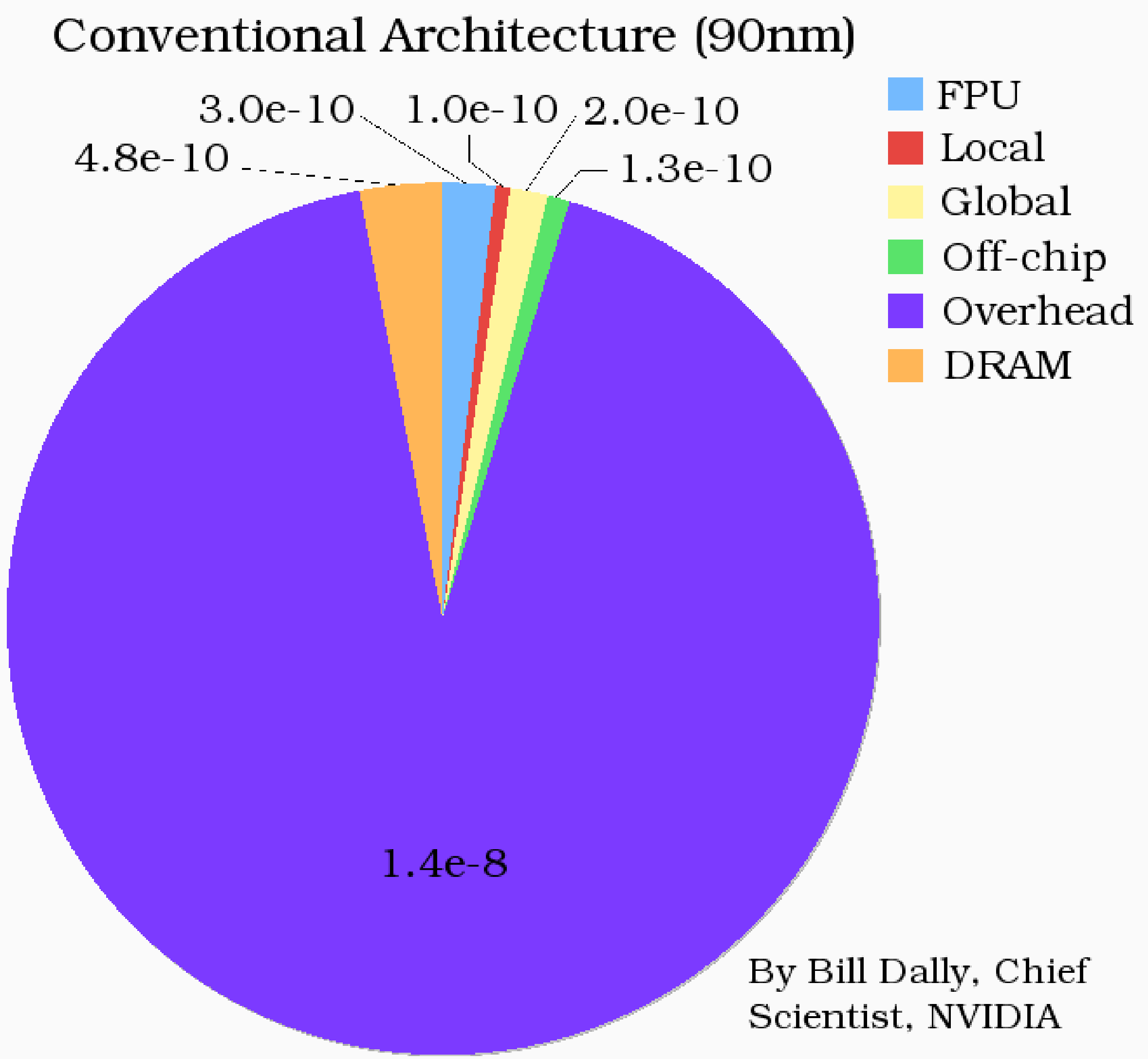
400x improvement in Flops per Watt!

Rise of Many-core Systems

Focus is on Flops per Watt:

- Clock rates constant or decreasing.
- Use larger fraction of transistors for calculation, split into many “throughput” cores.
- Explicit memory hierarchy. Cache management now in software stack. RAM/node $\uparrow$, but RAM/core $\downarrow$

CPU Power Consumption



By Bill Dally, Chief Scientist, NVIDIA

Rise of Many-core Systems

Focus is on Flops per Watt:

- Clock rates constant or decreasing.
- Use larger fraction of transistors for calculation, split into many “throughput” cores.
- Explicit memory hierarchy. Cache management now in software stack. RAM/node $\uparrow$, but RAM/core $\downarrow$
- Keep some traditional “low latency” cores around for coordination.
- Filesystem I/O even more of a bottleneck...

"Throughput" Processors

- NVIDIA Fermi: 480 cores @ 700MHz
- ATI Radeon 5970: 3200 cores @ 725MHz
- Intel Many Integrated Core (MIC): re-brand of failed Larrabee platform...
- Goal is to use something closer to 25% of transistors for Flops.
- Requires fine-grained parallelism, explicit memory movement.

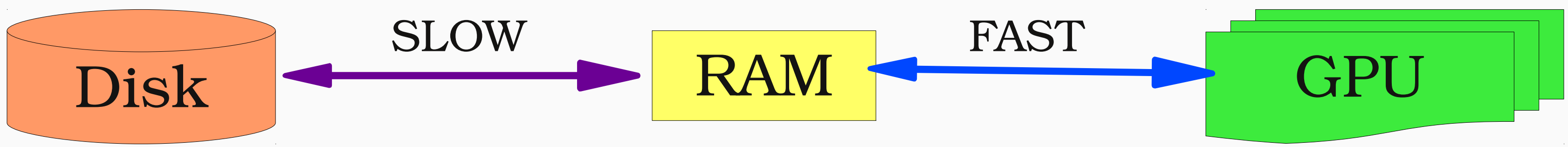
What does this mean for Astrophysics?

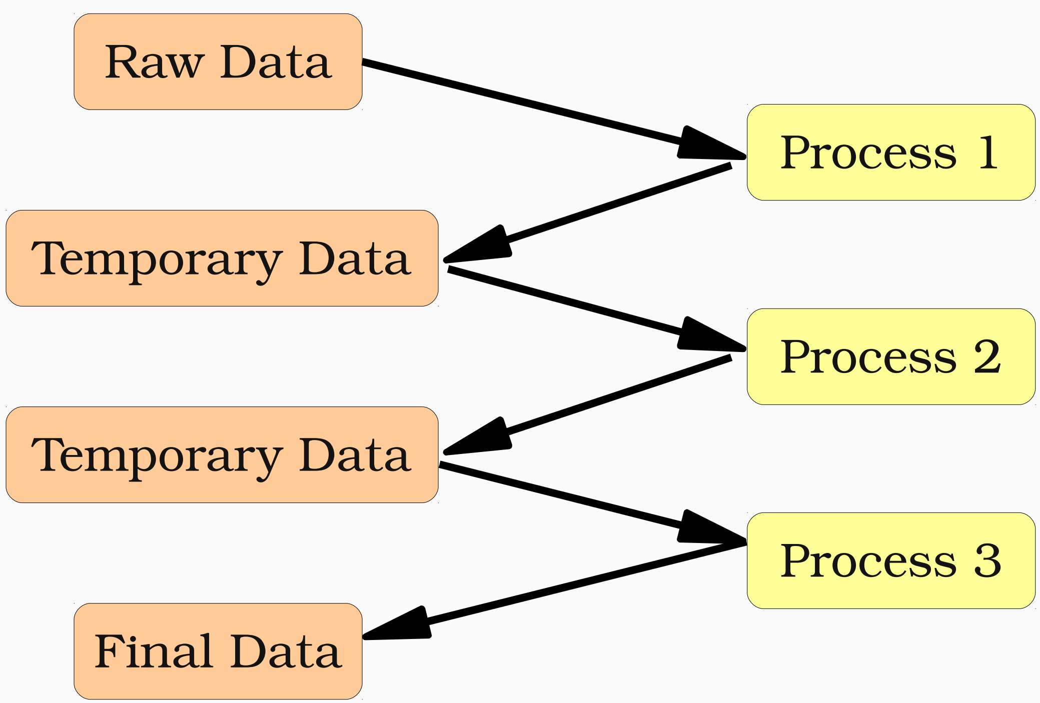
- Astrophysical datasets are getting larger!
  - LSST: 15TB / day
  - Near-term CMB missions: O(100-1000) TB
- Systems in the very near future may have O(10) traditional cores and O(100-1000) throughput cores per node.
  1. Data movement can be more costly than calculations- minimize when possible.
  2. Determine what operations can be parallelized at the node level.
  3. Evaluate new tools as they become available.

Data Movement

- Traditional paradigm:
  - Many small executables chained together
  - Write / read intermediate files
- This breaks down if:
  - I/O cost outpaces calculation AND
  - Overall runtime is unacceptably slow
- Movement to/from accelerators can also cancel benefit for some algorithms.

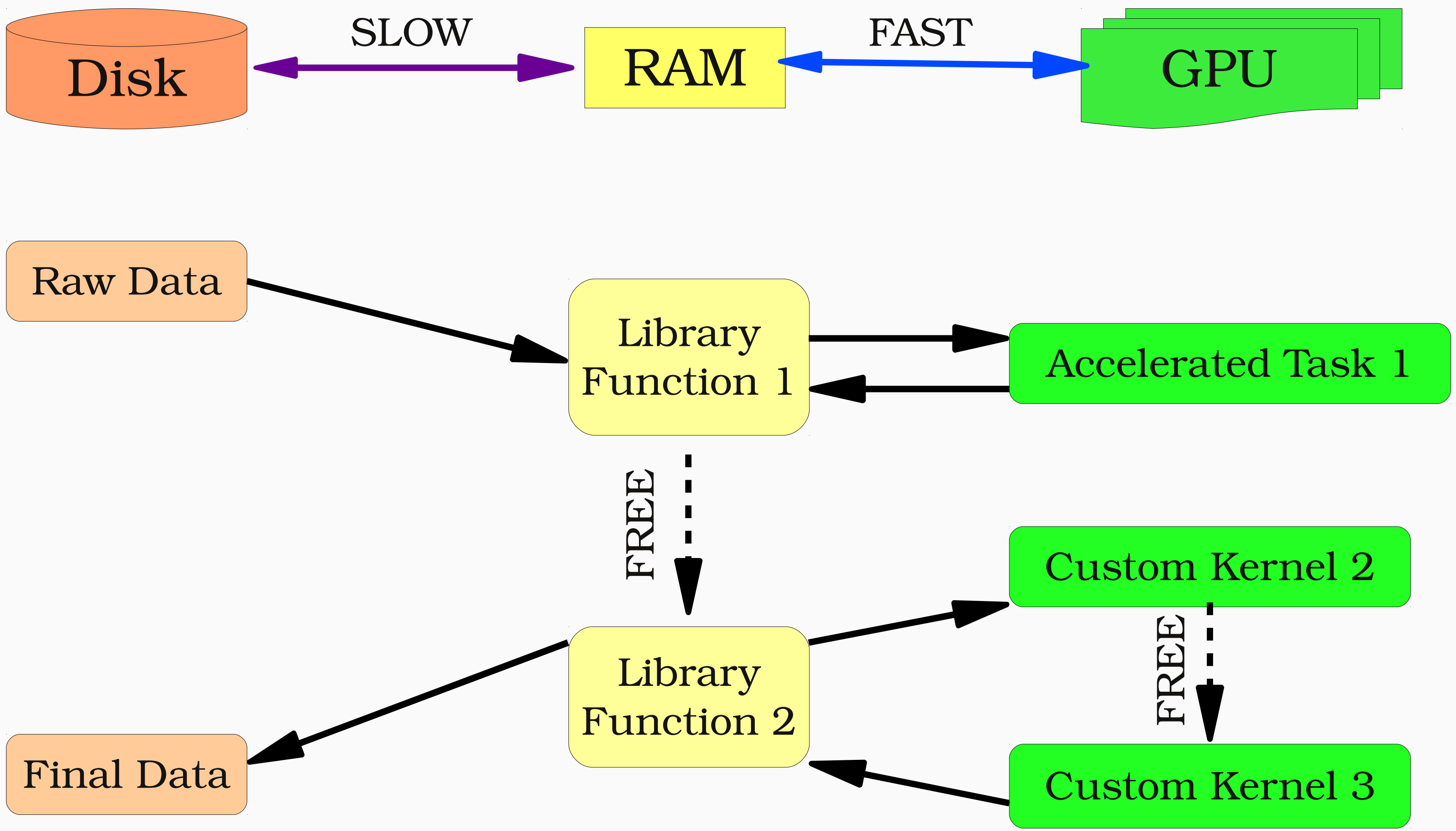
Data Movement for "Chained Processes"





This is for playing Quake, right?

Improved Data Movement



Parallelize Relevant Operations

- Split processing based on independent data products (embarrassingly parallel work flows)
- 1D – time domain astrophysics:
  - vector math, FFTs, sparse matrix operations.
- 2D – image / map manipulation
  - Linear combinations, projections
  - convolution / filtering, spherical harmonic transforms
- 3D – data cube (spaxel/voxel) manipulations.

Parallelize Relevant Operations

- Start by converting/switching low-level libraries
  - Likely to get some improvement without much work, e.g. FFT libraries.
- Only build custom code when needed- if data movement to/from card is dominant.
  - Use helper tools: PGI accelerator framework, MOAT (shameless plug!).

New Tools

- We are faced with a huge diversity of platforms: GPUs/accelerators from different vendors, varying OS support.
- OpenCL: Unified interface to CPU/GPU devices, wide industry support.

Conclusions

1. Start planning now for future hardware: will your code be ready for the cluster you purchase in 3 years?
2. Start testing new software tools that seem promising- what pieces of existing code are easy to parallize?
3. Will your future data volume overwhelm your current I/O patterns?