Challenges at Petascale for Pseudo-spectral Methods on Spheres (A Last Hurrah?)

Tom Clune
Software Integration and Visualization Office
NASA GSFC

Gone The Way of the Dinosaur?

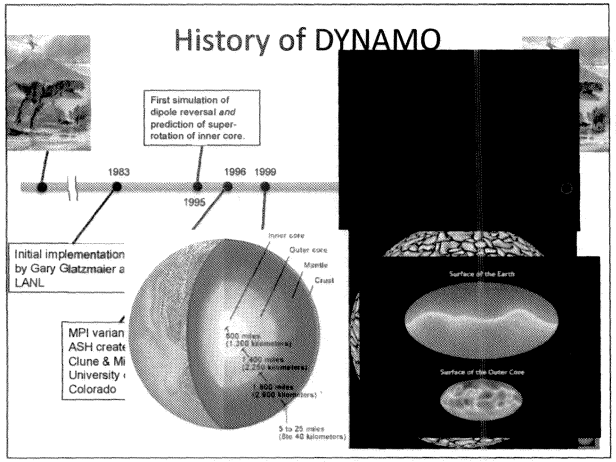
Nice Mathematical Properties

- "Exponential" accuracy
  - Double the effective resolution compared to FD
  - Commonly used as baseline for comparison
- Fast inversion of elliptic operators
  - Diagonal or nearly diagonal matrices
  - Enables efficient implicit time stepping
- Natural boundary conditions

PS Weaknesses

- Poor quality near discontinuities – e.g. terrain
- Numerically expensive at high resolution – $O(n^4)$
- Heavy data movement – limited by bandwidth of the interconnect
- Must respect symmetry. E.g. implicit coriolis

Not immediately obvious where the tradeoffs fall.



Pseudospectral Refresher

DYNAMO Goals

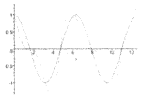
- Existing runs are for n ~ 500-1000
  - Run on variety of clusters
  - Performance constrained by 1D decomposition
    - O(500 nodes)
    - Wasting lots of cores to get necessary memory
    - Limited to ~ 1 TF
- Would like to achieve ~10x resolution
  - Needs **at least** petascale platform
  - Need consistent 2D (or even 3D) domain decomp
- Stretch – implicit treatment of coriolis

Pseudospectral Methods

Complementary mathematical representations are used in different phases of calculations.

Spatial derivatives computed in frequency (spectral) domain:

$$\nabla^2 A(\mathbf{k}_{ijk}) = |\mathbf{k}_{ijk}|^2 A(\mathbf{k}_{ijk})$$



Nonlinear products evaluated in spatial domain ("configuration space"):

$$f_{ijk} = b_{ijk} * u_{ijk}$$

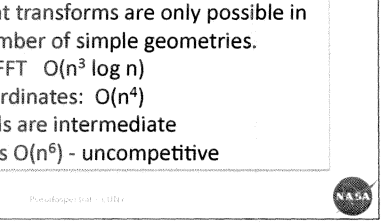
Each type of calculation is extremely **simple** ***and*** **accurate** in the appropriate domain.

Spatial and spectral domains are connected via *transforms*:

$\Phi(\mathbf{k})$ ⟷ transform ⟷ $\phi(\mathbf{x})$

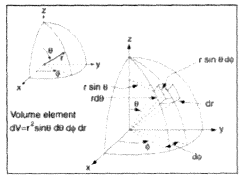
Simple and efficient transforms are only possible in relatively small number of simple geometries.

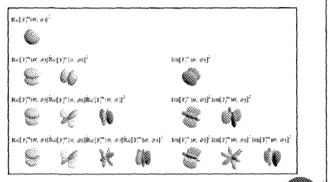
- Periodic box: FFT $O(n^3 \log n)$
- Separable coordinates: $O(n^4)$
- Spherical shells are intermediate
- General case is $O(n^6)$ - uncompetitive



PS Methods on Spherical Shells

- Spectral expansion based upon spherical harmonics
  - $Y_{lm}(\theta,\phi)$ $(m = 0, \ldots, m_{max}, l = 0,\ldots,l_{max}(m))$
- Radial expansion based upon Chebyshev polynomials
  - $T_n(r)$ $(n = 0, \ldots, n_{max})$
- Poisson operators block **diagonal** (nontrivial coupling in radial)

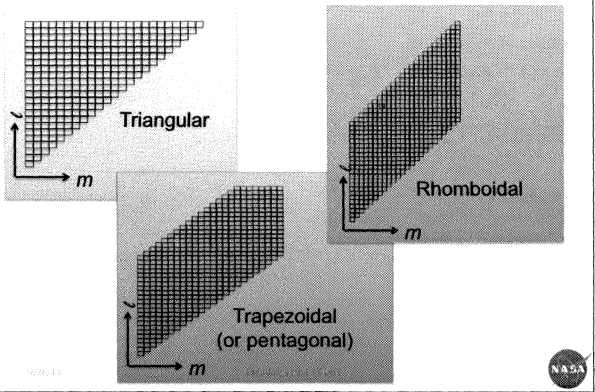




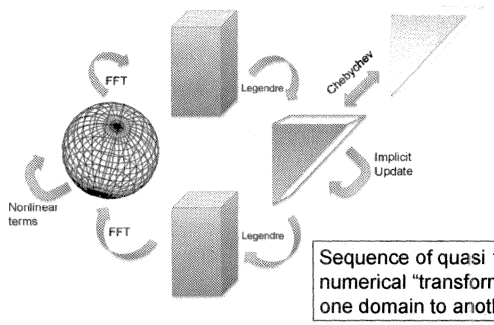
Interesting Software Aspects

- Elegant program structure
  - Sequence of transforms coupled by memory transposes
  - Software *infrastructure* plays major role
    - Non-rectangular domains
    - Non-trivial domain decomposition
    - Non-*obvious* data layout
- Unique performance aspects
  - Different scaling properties: transpose vs. halo fill
  - Nothing to optimize!
    - FP workload largely in optimized libraries (FFT, DGEMM, ...)
    - All-to-all is part of HPC benchmarks

Spherical Harmonic Truncations



Sketch of PS Implementation



Sequence of quasi 1D numerical "transforms" from one domain to another.

Constraints on Decomposition

- General anatomy of a transform
  - **Acts** on 1 axis of domain Q(i,j,k) -> Q'(m,j,k)
  - **Independent** of at least one axis
  - Possibly **parameterized** by another axis
- Example: Legendre Transform
  - Acts on meridional coordinate/degree
  - Independent of radial coordinate
  - Parameterized by wavenumber coordinate

Domain Decomposition Strategy For a Given Transform

- **Co-locate** transform axis
  - Perform operation "in processor"
  - Leverage serial implementation
  - Often available in optimized library (FFT, DGEMM)
- **Distribute** remaining coordinates
  - No computational dependencies
  - Effective 2D decomposition
- **Balance/Optimize**
  - Memory/performance optimization by grouping along "independent" axis.

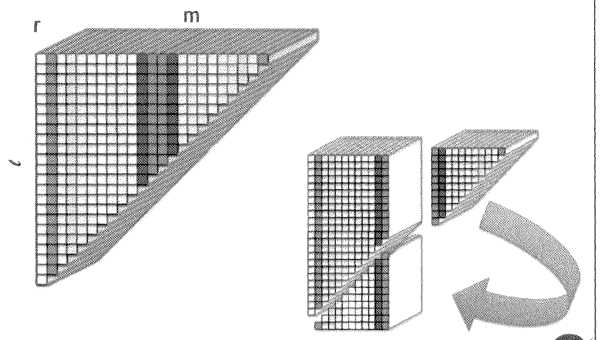
Unfortunately …

- No decomposition scheme works for all cases
  - Usually need separate layout for *each* transform!
  - Sometimes can do 2 transforms on one layout if using a 1D decomposition
- Load balance is nontrivial for non-rectangular domains.

Example: Legendre Transform

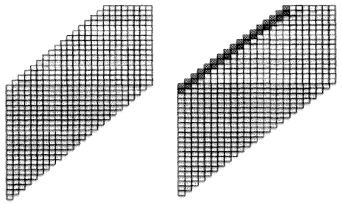
- Transforms degree ℓ to/from θ (DGEMV)
  - Independent of radial coordinate **r**
  - Parameterized by azimuthal wavenumber **m**
- Decomposition constraints:
  - Keep ℓ and θ in processor
  - Distribute wavenumbers across processors.
  - Group **r** on processor to improve cache reuse (DGEMV -> DGEMM)
    - Split blocks of **r** across processors to balance scalability against serial performance.

Load Balance and Triangular Domains



Implicit Update

- Couples radial coordinate (DGETRS)
  - Mostly independent of azimuthal wavenumber
  - Dependent on degree λ
- Strategy
  - Group **wavenumber** for BLAS2 -> BLAS3



Spectral Framework

SPF

Abstractions

Axis: Label(s) and coordinate indices

R
0
1
2
...

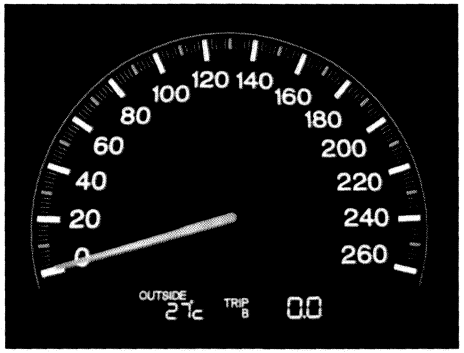
LM	L	M
0	0	0
1	0	1
2	1	1
...		

Triangular domain treated as 1D

Domain: Product of axes
- Note, collection of fields can be an "axis"
- DistributedDomain: subclass with "map"

Field: Domain reference with array of values

Going to Petascale



Abstractions (cont'd)

Transformer:
- Produces: Decomposition constraints, Cost function
- $Field_{in}$ → $Field_{out}$

Balancer:
- Ingest: Transformer, Auxiliary Coords, Communicator
- Produces: In/Out Domains, Instantiated Transformer

Transposer:
- Initialization: {$Domain_A$, $Domain_B$}
- $Field_{in}$ → $Field_{out}$

Specific Challenges

- Accuracy
- Transposes
- Initialization

Representative Numbers

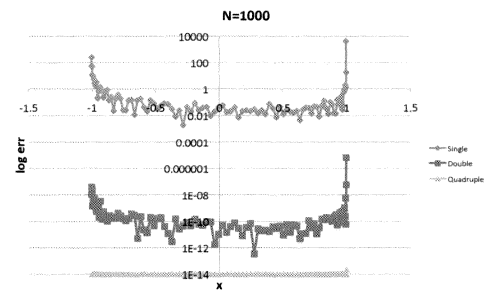
- Problem Size

Parameter	Value
n – grid pts in 1 dimension	10^4
N – total grid points	10^{12}
N_t – number of iterations	10^6
N_l – Legendre per step	30

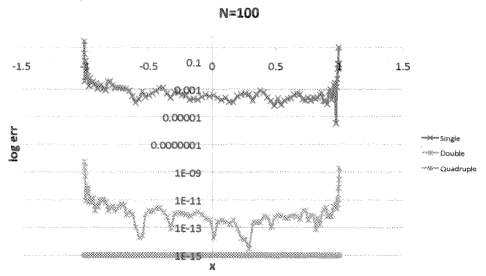
- R=10 PF (10^{16} FP/s)

Parameter	Intel + IB	"Blue Waters"	"Blue Gene"
R_c – core peak (GF)	20	32	12
N_p - # proc.	500,000	350,000	800,000
N_c - # core/node	32	32	16
N_n – # nodes	16,000	11,000	50,000
α – network latency (μs)	1?	1?	1?
B – bisection bw (TB/s)	1?	500	20

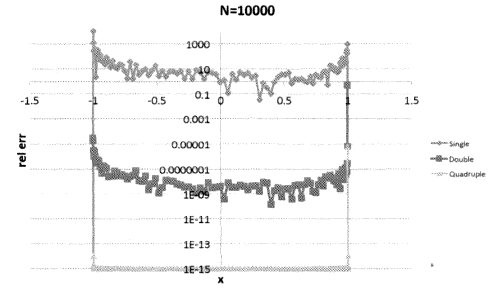
Legendre 2nd Derivative



Legendre 2nd Derivative



Legendre 2nd Derivative

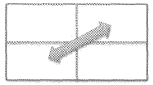


Back of the Envelope

- Compute time dominated by Legendre DGEMM:

$$T_{comp} \approx \frac{N_L N_t n^4}{R^{eff} p}$$

- Communication dominated by global transposes
  - Each transpose moves ¼ of data to other half of machine



  - Each process sends packet to $p^{1/D}$ other processes (D=1,2,3)

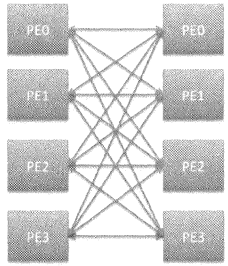
$$T_{comm} \approx N_l N_t \left(\alpha^{eff} p^{1/D} + \frac{4n^3}{B^{eff}} \right)$$

Desirable Characteristics

- Domination by computations: $n > \frac{4Rp}{B}$
  - 8 - 4000
- Bandwidth dominates latency: $n < \left(\frac{\alpha B p^{1/D}}{4} \right)^{1/3}$
  - 250 - 125000
- Complete in T_{sim} seconds: $n < \left(\frac{T_{sim} R_{eff} p}{N_l N_t} \right)^{1/4}$
  - 4000 for 1 week turnaround
- Efficient level 3 BLAS: $n >> p^{1/D}$
  - ~10000

Generalized Redistribution

Distribution 1 Distribution 2



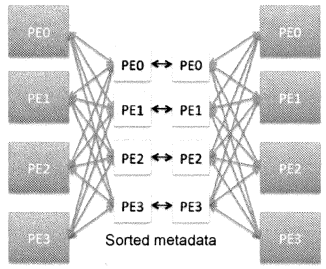
Naïve Initialization:
Each processor exchanges ***all*** metadata with all others and determine overlapping elements. Complexity is $O(N^2)$

Consider a petascale problem:
$N=10^{12}$ (10^4 pts each axis)
10 peta-op platform

At 10 ops/comparison and 10% efficiency, initialization requires 10^{10} sec. or 300 years!

Fast Initialization

Distribution 1 Distribution 2



1. Global (sample) sort is used to order metadata.
   - Complexity ~$O((N/p) \log_2 p)$
   - Use same pivots for both distributions.
   - Data tagged with original PE
2. Each processor compares sorted metadata to determine overlap.
3. Overlap data is then "unsorted".

Petascale initialization should require ~10 minutes.

PGAS (CAF)

- Efficient implementations of realistic global transposes are intricate and tedious in MPI.
- PS at petascale requires exploration of a variety of strategies for spreading local and remote communications.
- PGAS allows far simpler implementation and thus rapid exploration of variants.

Conclusions

- Proper software abstractions should enable rapid-exploration of platform-specific optimizations/ tradeoffs.
- Pseudo-spectral methods are marginally viable for at least some classes of petascale problems.
  - A GPU based machine with good bisection would be best.
- *Scalability* at exascale is possible, but the necessary resolution will make algorithm prohibitively expensive.