BERKELEY LAB
LAWRENCE BERKELEY NATIONAL LABORATORY

U.S. DEPARTMENT OF ENERGY

Computer Science & Performance Evaluation

Erich Strohmaier,

Lawrence Berkeley National Laboratory

Large Scale Computing and Storage Requirements for
Advanced Scientific Computing Research
ASCR / NERSC Workshop

January 5-6, 2011

Some Current Projects

- UPC, CAF and Titanium
  - And hybrids of these with others (MPI)
- Performance Characterization and Benchmarking of HPC Systems (Apex-MAP)
  - Synthetic parameterized performance probes
- The Performance Engineering Research Institute (PERI)
  - Application centric performance engineering
- Developing and optimizing new algorithms
  - Cache - Math/CS Institute
- (Evaluation of) of new and of hybrid programming models
- Various other benchmarking, auto-tuning, and application optimization studies

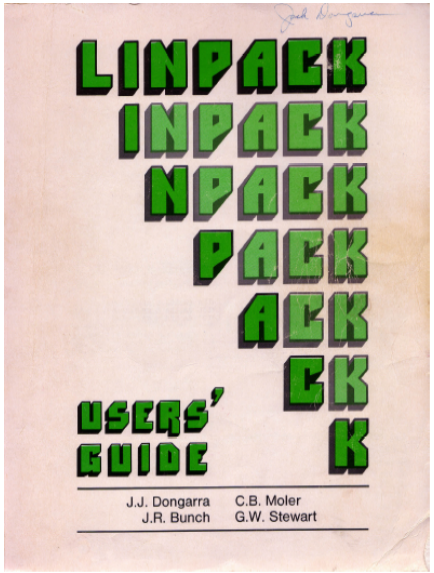
The Accidental Benchmarker

- Appendix B of the Linpack Users' Guide
  - Designed to help users extrapolate execution for Linpack software package
- First benchmark report from 1977;
  - Cray 1 to DEC PDP-10

UNIT = 10**6 TIME/(1/3 100**3 + 100**2)

Facility	TIME N=100 secs.	UNIT micro- secs.	Computer	Type	Compiler
NCAR	.049	0.14	CRAY-1	S	CFT, Assembly BLAS
LASL	.148	0.43	CDC 7600	S	FTN, Assembly BLAS
NCAR	.192	0.56	CRAY-1	S	CFT
LASL	.210	0.61	CDC 7600	S	FTN
Argonne	.297	0.86	IBM 370/195	D	H
NCAR	.359	1.05	CDC 7600	S	Local
Argonne	.388	1.33	IBM 3033	D	H
NASA Langley	.489	1.42	CDC Cyber 175	S	FTN
U. Ill. Urbana	.506	1.47	CDC Cyber 175	S	Ext. 4.6
LLL	.554	1.61	CDC 7600	S	CHAT, No optimize
SLAC	.579	1.69	IBM 370/168	D	H Ext., Fast mult.
Michigan	.631	1.84	Amdahl 470/V6	D	H
Toronto	.890	2.59	IBM 370/165	D	H Ext., Fast mult.
Northwestern	1.44	4.20	CDC 6600	S	FTN
Texas	1.93*	5.63	CDC 6600	S	RUN
China Lake	1.95*	5.69	Univac 1110	S	V
Yale	2.59	7.53	DEC KL-20	S	F20
Bell Labs	3.46	10.1	Honeywell 6080	S	Y
Wisconsin	3.49	10.1	Univac 1110	S	V
Iowa State	3.54	10.2	Itel AS/5 mod3	D	H
U. Ill. Chicago	4.10	11.9	IBM 370/158	D	G1
Purdue	5.69	16.6	CDC 6500	S	FUN
U. C. San Diego	13.1	38.2	Burroughs 6700	S	H
Yale	17.1*	49.9	DEC KA-10	S	F40

* TIME(100) = (100/75)**3 SGEFA(75) + (100/75)**2 SGESL(75)



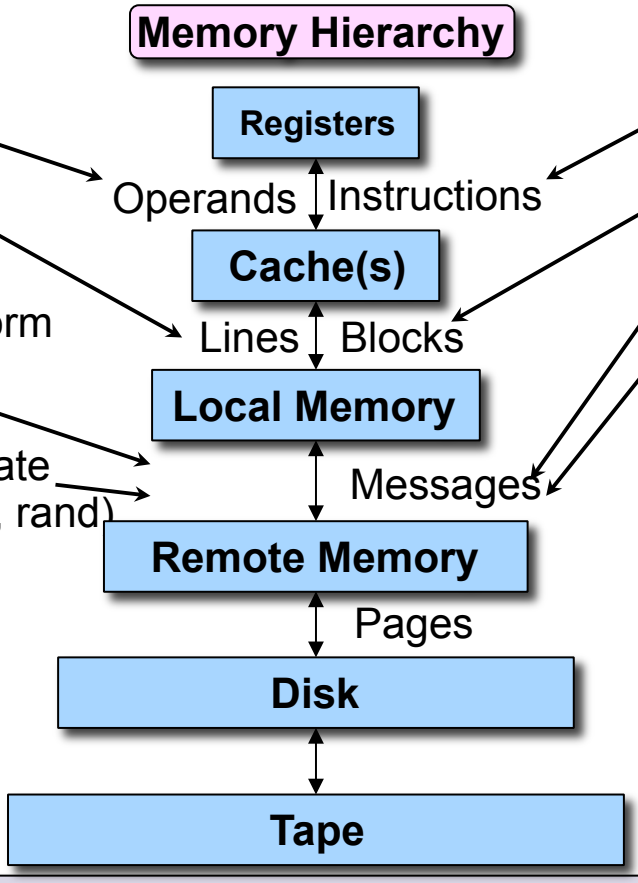
Dense matrices
Linear systems
Least squares problems
Singular values

From: J.J. Dongarra

HPCS Performance Targets

- HPL: linear system solve
  Ax = b
- STREAM: vector operations
  A = B + s * C
- FFT: 1D Fast Fourier Transform
  Z = fft(X)
- RandomAccess: integer update
  T[i] = XOR(T[i], rand)

HPC Challenge



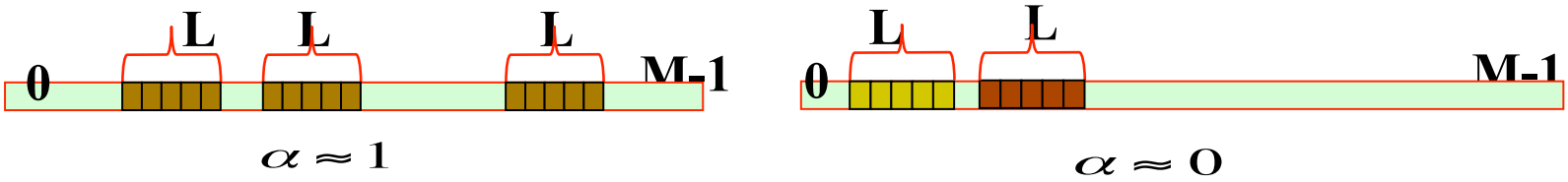
Max	Relative
2 Pflop/s	8x
6.5 Pbyte/s	40x
0.5 Pflop/s	200x
64000 GUPS	2000x

Performance Targets

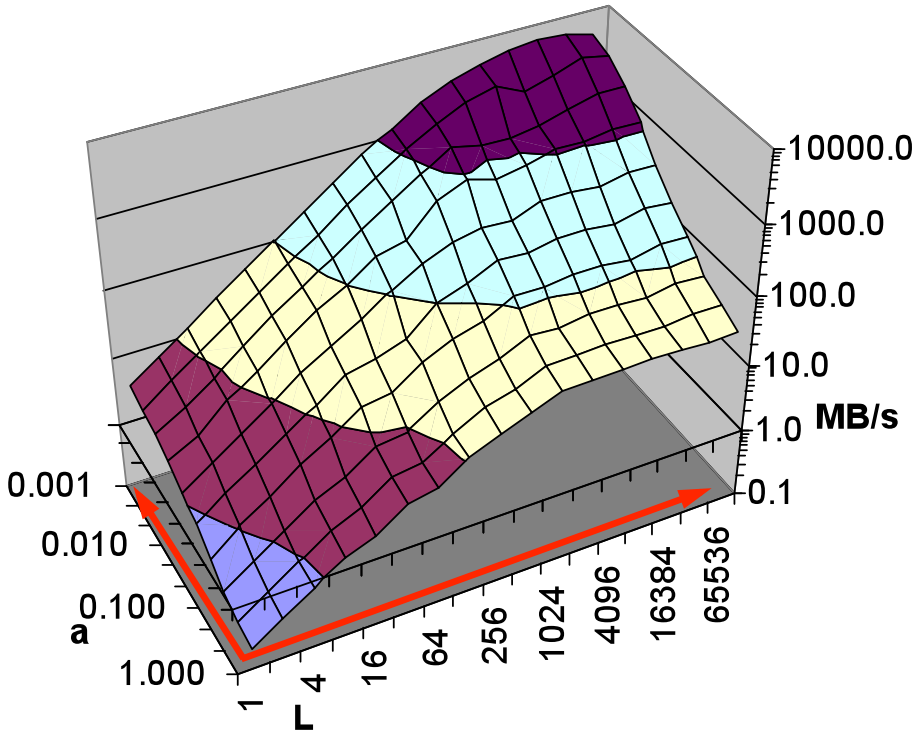
- **HPCC was developed by HPCS to assist in testing new HEC systems**
- **Each benchmark focuses on a different part of the memory hierarchy**
- **HPCS performance targets attempt to**
  - **Flatten the memory hierarchy**
  - **Improve real application performance**
  - **Make programming easier**

APEX-Map: Locality Concepts

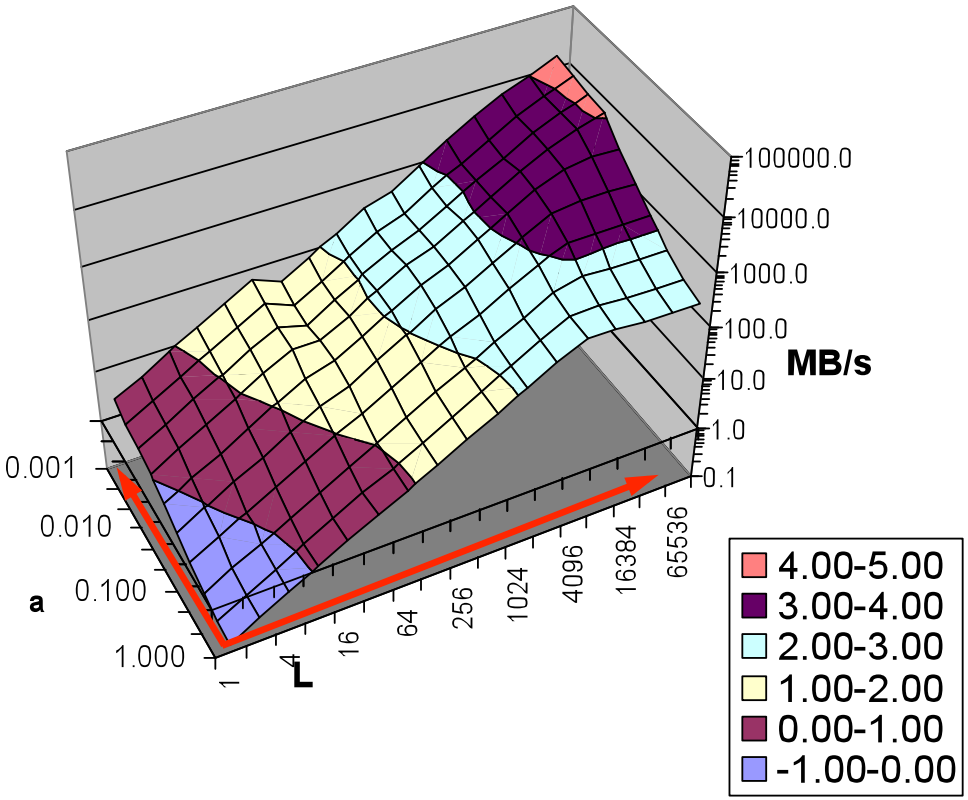
- Data set size: ***M***
- Spatial Locality (***L***):
  - Blocked access to *L* contiguous data elements.
  - *L* is also the innermost loop length!
- Temporal Locality (α):
  - Achieve more frequent access to certain memory locations by using **non-uniform random** starting addresses of blocks distributed according to a power law.
  - Characterize temporal locality with the exponent α of the power law (α in [0,1]).



CRD Parallel Performance Surfaces 256 Processors - MPI



Cheetah – IBM SP Power4

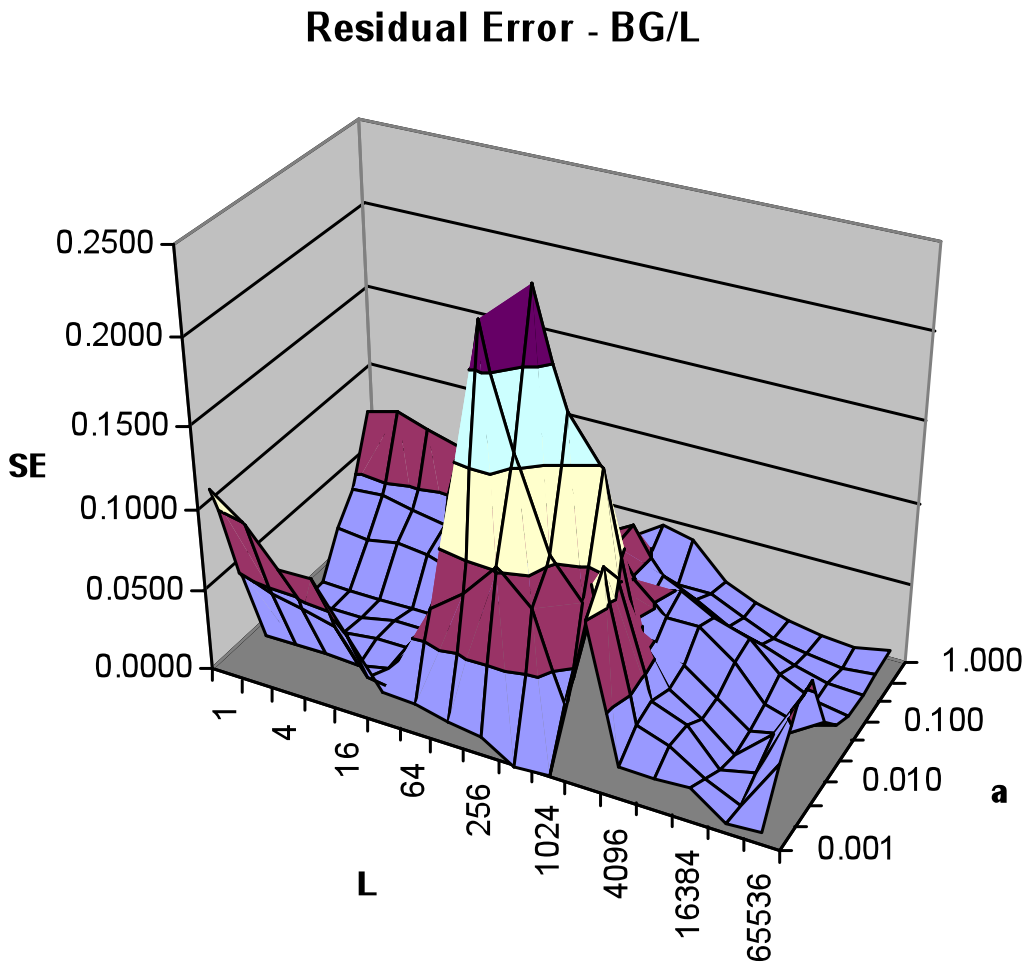


Phoenix – Cray X1

Performance Model

- **Linear timing for two levels**
  - *T = [P(c/m)*(a+b*(L-1)) + (1-P(c/m))*(c+d*(L-1))]/L*
    - *P(c/m): Local access probability*
    - *a= local latency;*
    - *b= local gap;*
    - *c= remote latency;*
    - *d= remote gap;*
- **Characterize systems with 5 parameters!**
- **Use performance models to eliminate the 'expected' performance behavior of APEX-Map:**

Residual Error - Parallel



TORCH: Computational Reference Kernels

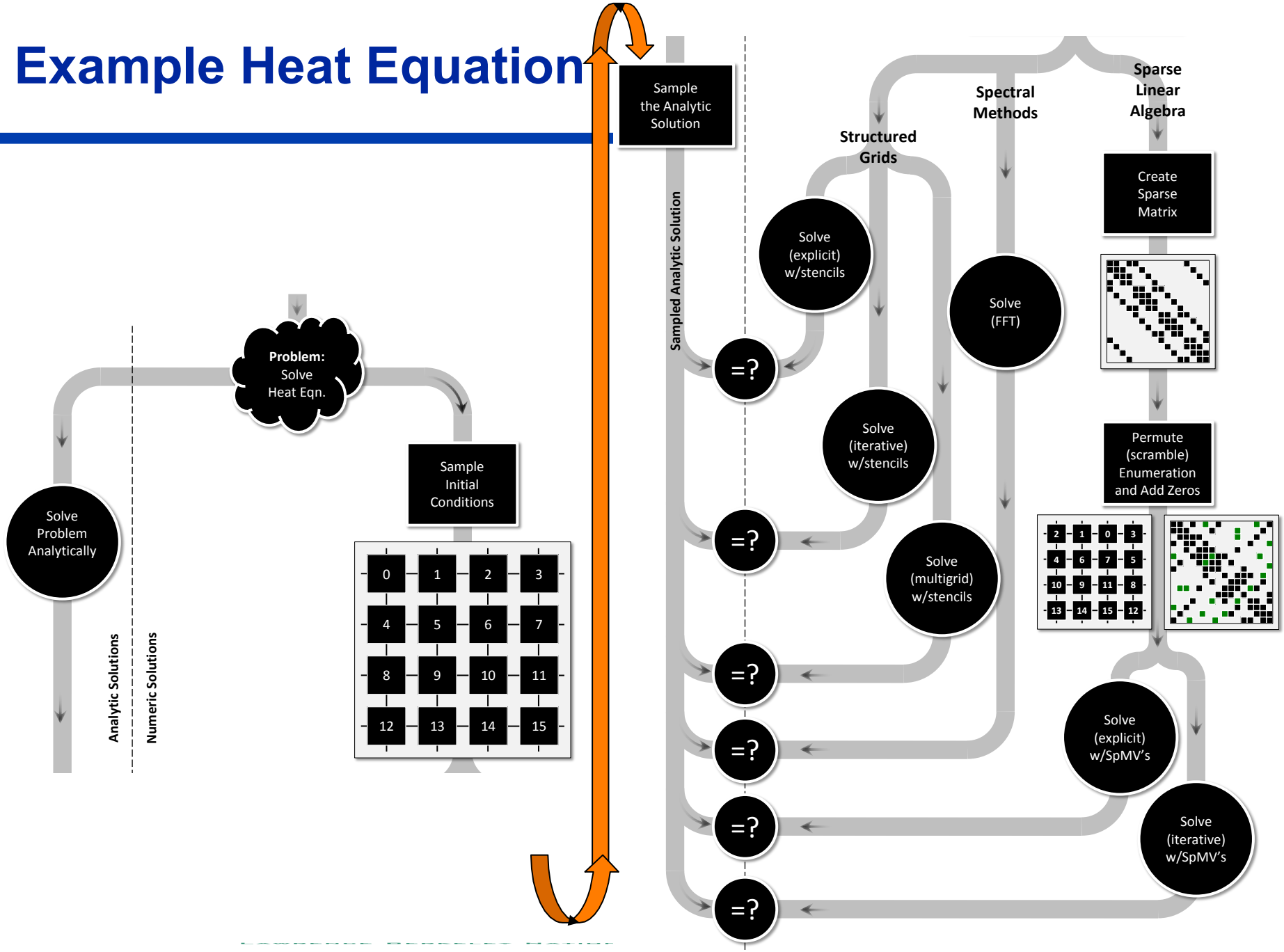
Kernel	Dense Linear Alg.	Sparse Linear Alg.	Structured Grids	Unstructured Grids	Spectral	Particles	Monte Carlo	Graphs & Trees	Sort	Definition	Reference	Optimized	Scalable Inputs	Verification
Scalar-Vector Mult.	■									✓	✓		✓	
Elementwise-Vector Mult.	■									✓	✓		✓	
Matrix-Vector Mult.	■									✓	✓		✓	
Matrix-Matrix Mult.	■									✓	✓		✓	
LU Factorization	■									✓	✓	✓	✓	✓
Symmetric Eigensolver (QR)	■									✓	✓		✓	✓
Cholesky Factorization	■									✓			✓	
SpMV (y=Ax)		■								✓	✓	✓	✓	✓
SpTS (Lx=b)		■								✓	✓		✓	✓
Matrix Powers (y_k=A^kx)		■								✓	✓		✓	✓
Solve PDE via CG	■	■								✓	✓		✓	✓
Solve PDE via KSM/GMRES	■	■								✓	✓		✓	✓
Solve PDE via SpLU	■	■												
Finite Difference Derivatives			■							✓	✓		✓	✓
FD/Laplacian			■							✓	✓	✓	✓	✓
FD/Gradient			■							✓	✓	✓	✓	✓
FD/Divergence			■							✓	✓	✓	✓	✓
FD/Curl			■							✓	✓		✓	✓
Solve FD/PDE (explicit)			■							✓	✓		✓	✓
Solve FD/PDE via CG	■		■							✓	✓		✓	✓
Solve FD/PDE via Multigrid	■		■							✓	✓		✓	✓

There are a number of other important structured grid methods including lattice Boltzmann, finite volume, and AMR, that we have yet to enumerate representative kernels for.

TORCH: Computational Reference Kernels

Kernel	Dense Linear Alg.	Sparse Linear Alg.	Structured Grids	Unstructured Grids	Spectral	Particles	Monte Carlo	Graphs & Trees	Sort	Definition	Reference	Optimized	Scalable Inputs	Verification
Although even within our community unstructured grids are commonly used, we have yet to enumerate any concise representative kernels.														
1D FFT (complex→complex)										✓	✓		✓	✓
3D FFT (complex→complex)										✓	✓		✓	✓
Convolution										✓	✓		✓	✓
Solve PDE via FFT										✓	✓		✓	✓
2D N^2 Direct										✓	✓		✓	
3D N^2 Direct										✓	✓		✓	
2D N^2 Direct (with cut-off)										✓	✓		✓	✓
3D N^2 Direct (with cut-off)										✓	✓		✓	✓
2D Particle-in-Cell (PIC)														
3D Particle-in-Cell (PIC)														
2D Barnes Hut										✓	✓		✓	✓
3D Barnes Hut										✓	✓		✓	✓
2D Fast Multipole Method														
3D Fast Multipole Method												✓	✓	
Quasi-Monte Carlo Integration										✓	✓		✓	✓
Breadth-First Search										✓	✓		✓	✓
Betweenness centrality										✓	✓		✓	✓
Integer Sort										✓	✓		✓	✓
100 Byte Sort										✓	✓		✓	✓
Spatial Sort										✓	✓		✓	✓

Example Heat Equation



Past and Current Use

- A typical study with a few large scale application runs takes between 100k to several M CPU core-hours (spread over multiple systems)
- Language and communication software developed on small to medium systems with occasional large scale tests (several 100k core-hours)
- Auto-tuning focuses on node-level issues
  - –diversity and ease of access more important than time
- Disk required for some trace-files
  - –Not often on large scale runs (too long, too big)

What is Changing?

- Increasing diversity of architectures and programming models
  - –No single, unified targets for studies on the horizon
  - –Explorative evaluation studies need to consider lot more ‘cases’ (Kernels, codes, implementations, architectures)
- Increasing complexity of single architectures
  - –Evaluation requires parameter sweeps with probes
  - –Optimization requires auto-tuning, which becomes a search problem in large spaces

What is Changing?

- 10× performance in 3 years drives concurrency levels
  - –Plus any changes driven by architectures
- Concurrency level will grow more rapid than in the past
  - –Scalability questions more pressing as we go forward
  - –More large scale experiments needed
- More research groups will hit problems
  - –More performance studies and work overall
- Drive to Exascale will increase need for large scale performance studies, simulation, co-design, and development in general

Coming Changes

- More focus on large scale scalability, multiple implementations, and larger variety of architectures will increase demand for CPU core-hours for most(!) studies.
- Language and communication software development needs to focus on large scale issues
- The search space for auto-tuning on the node-level increases and more studies will look at interaction of larges scale MPI with local xyz optimizations.
- CoDesign will place new demands
- Disk and I/O requirements could easily explode if large scale tracing, debugging, and simulation are necessary