Parallel Fast Fourier Transform Libraries

Benson Muite with help and support of many others!

benson.muite@ut.ee
http://kodu.ut.ee/~benson
http://parallel.computer

20 February 2017

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

A Disclaimer

- Work presented is not all my own
- Views stated may not be shared by all those who have contributed

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Outline

- Benchmarking of high performance computers
- Linpack
- Scientific computing and procurement - the Poisson equation
- Stream
- Conjugate Gradients - HPCG
- Multigrid - HPGMG
- FFT
- Fast Fourier Transform libraries
- Klein Gordon equation as a benchmark
- Demo

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Benchmarking of high performance computers

- Partly a sport
- Indicate opportunities for algorithm implementation optimization
- Indicate opportunities for improved algorithm choice
- Method of providing guidance on performance and in finding errors
- Allows for reasonable comparisons between very different platforms

Benchmarking of high performance computers

- Benchmarks usually driven by aspects that limit problem solving effectiveness
- Not all will be equally important for you
- What would be a minimal set of benchmarks that would be useful for a wide variety of centers to run?

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

The stream benchmark

- Main observation is that memory bandwidth is limiting factor for performance rather than floating point operations
- Examine performance of systems when doing vector products

$$A = A + B * C$$

- Three floating point operations per memory access
- Micro benchmark for single processor, but for many parallel computations, it is more relevant than communication between nodes

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Graph 500

- http://www.graph500.org/
- Random access benchmark – breadth first search
- Computer network is main limitation

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Linpack benchmark

- Solve dense system of linear equations

$$Ax = b$$

- Main kernel is dense matrix multiply from doing LU decomposition

$$LU = A$$

$$Ux = c$$

$$Lc = b$$

- Tuning and parallelization strategies can require a long time to get this right, even though dense matrix multiply us easier to tune
- Gives Top 500 list

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Theoretical peak performance - max flops

- Small microbenchmark
- Repeatedly compute

$$a = a + c * b$$

- All elements stored in registers, so no memory access limitations
- Take advantage of fused multiply add floating point instruction
- Easy to tune and port this to different architectures

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

HPCG

- High performance conjugate gradient
- Suppose we need to solve a linear system of equations

$$Ax = b$$

- $$A$$

  is typically sparse
- Solving using exact method as used in Linpack can have very high memory requirements
- Can also require many more floating point operations than other methods that might give “good enough” results
- Use iterative methods where dominant component is sparse matrix vector multiply
- As an example decompose the matrix

$$A = D + R$$

, where D is a matrix system that is easy to solve

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Conjugate Gradient Iteration

- Why does Conjugate Gradient Iteration work?
- Start with a simpler case, gradient descent!

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Steepest Descent for Linear Systems of Equations

- Solving $\mathbf{Ax} = \mathbf{b}$ is equivalent to minimizing $\mathbf{x^T A x} - \mathbf{x^T b}$ for symmetric and positive definite matrix $\mathbf{A}$
- the algorithm is
  - choose $\mathbf{x_0}$ so that $\mathbf{r_0} = \mathbf{Ax_0} - \mathbf{b}$
  - FOR $k = 1, 2, ..$

$$\alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_k^T \mathbf{A} \mathbf{r}_k}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{r}_k$$

$$\mathbf{r}_{k+1} = \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}$$

ENDFOR

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Conjugate Gradient Method

- Solving $\mathbf{Ax} = \mathbf{b}$ is equivalent to minimizing $\mathbf{x^T Ax} - \mathbf{x^T b}$ for symmetric and positive definite matrix $\mathbf{A}$
- **Theorem** Let $\mathbf{A}$ be a symmetric positive definite matrix of size $n \times n$. Then after n conjugate direction searches in the n dimensional space, we obtain $\mathbf{r_n} = \mathbf{0}$.
- The algorithm
  - choose $\mathbf{x_0}$ so that $\mathbf{p_o} = \mathbf{r_0} = \mathbf{Ax_0} - \mathbf{b}$
  - FOR $k = 1, 2, .., n$

$$\alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$$

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{p}_k$$

$$\beta_k = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}$$

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$$

ENDFOR

High Performance Conjugate Gradient

- https://software.sandia.gov/hpcg/html/index.html
- Solve a sparse matrix system $\mathbf{Ax} = \mathbf{b}$ using a prescribed iterative method
- Several mathematical techniques used to improve convergence of conjugate gradient method
- Main algorithmic components remain the same
- Method is sub optimal with respect to asymptotic operation count
- Method may allow for comparison of factors on machines other than floating point operations
- Still not fully reflective of supercomputer performance
- Detailed set of rules as to what performance tuning changes are allowed
- Does not allow for algorithmic improvements
- One study indicates a very strong correlation with stream benchmark

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Multigrid

- Iteration usually works well on high frequency errors
- Low frequency errors will take a long time to be eliminated
- To speed this up, solve an problem on a small grid, then add scales to the grid to improve the approximation
- Choice of discretization is also important in efficiency of this method

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

High Performance Geometric Multigrid

- `https://www.hpgmg.org/`
- Solve a sparse matrix system $\mathbf{Ax} = \mathbf{b}$ using an iterative method
- Method can be optimal with respect to asymptotic operation count
- Method may allow for comparison of factors on machines other than floating point operations
- Does not allow for algorithmic improvements

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Other benchmarks

- NAS Parallel Benchmarks https://www.nas.nasa.gov/publications/npb.html
- HPCC Challenge suite http://icl.cs.utk.edu/hpcc/
- Example procurement benchmarks https://asc.llnl.gov/CORAL-benchmarks/

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Scientific computing

- Solution of problems from science and engineering
- Typically rely on numerical linear algebra
- One of early drivers of high performance computing
- For a long time, floating point operations limited what could be simulated

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Differential equations as models for physical processes

- Models first encountered by applied mathematicians, engineers and physicists
  - Heat equation
  - Schrödinger equation
  - Wave equation
  - Poisson equation

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

The Heat Equation

- Used to model diffusion of heat, species,
- 1D
$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$
- 2D
$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$
- 3D
$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$$
- Not always a good model, since it has infinite speed of propagation
- Strong coupling of all points in domain make it computationally intensive to solve in parallel

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Linear Schrödinger Equation

- Used to model quantum mechanical phenomena and often appears in simplified wave propagation models
- 1D
$$i\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$
- 2D
$$i\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$
- 3D
$$i\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$$
- Looks like a heat equation with imaginary time
- Strong coupling of all points in domain make it computationally intensive to solve in parallel

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

The Wave Equation

- Used to model propagation of sound, light
- 1D
$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2}$$
- 2D
$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$
- 3D
$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$$
- Has finite speed of propagation
- Finite signal propagation speed sometimes useful in parallelization since there is no coupling between grid points that are far apart, hence smaller communication requirements
- If propagation speed is very fast, communication requirements still important

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

The Poisson Equation

- Used to model static deflection of a drumhead
- 1D

$$f(x) = \frac{\partial^2 u}{\partial x^2}$$

- 2D

$$f(x,y) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

- 3D

$$f(x,y,z) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$$

- No time dependence, but also arises in time discretizations of time dependent partial differential equations

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Main role of computation

- Provide approximate solutions since exact ones typically not available
- High performance computing typically allows solution of more accurate models or ensemble exploratory simulation
- Has historically been seen as “science enabler” not real science
- This becomes problematic when software development time becomes a large part of a project

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

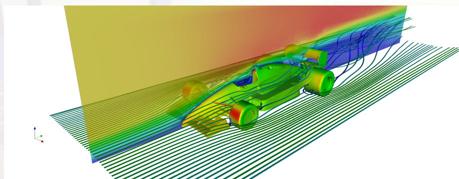
Typical scientific computing workflow

- Obtain model
- Find an approximation of this model suitable for computer simulation - best choice is problem and computer architecture specific
- Simulate the model
- Make sense of results from simulation - IO and visualization are key for this, but traditionally neglected
- Hope for benchmark information that reflects this workflow

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Aim to simulate this,

- https://www.flickr.com/photos/kitware/2293740417/in/pool-paraview/

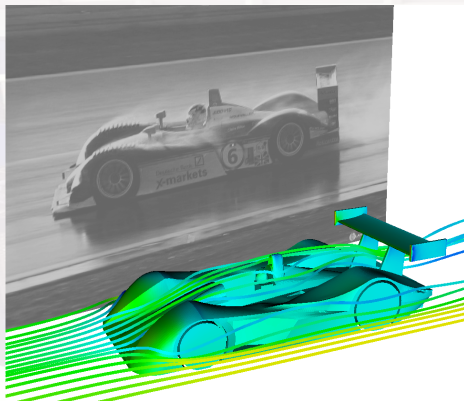


- Visualization around Formula 1 Race Car by Renato N. Elias, Rio de Janeiro, Brazil

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

or this

- https://www.flickr.com/photos/kitware/2294528826/in/pool-paraview/



- Visualization around Formula 1 Race Car by Renato N. Elias, Rio de Janeiro, Brazil

Maybe even this

- https://youtu.be/FTlJ9l9KTkw
- https://youtu.be/MsDgw8_cb90
- https://youtu.be/Su8qzC4HHVs

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmLl216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

The Heat Equation: Numerical Solution Methods

- Finite Difference
- Finite Volume
- Finite Element
- Spectral

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Finite Difference Method

- Approximate derivatives by difference quotients
- Simple method to derive and implement
- Convergence rates tend not to be great
- Difficult to use for complicated geometries
- Tends to scale well since communication requirements are low

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Finite Difference Method for Heat Equation

- $u_t = 8u_{xx}$
- Using backward Euler time stepping:

$$\frac{u_i^{n+1} - u_i^n}{\delta t} = 8\frac{u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}}{(\delta x)^2}$$

- Using forward Euler time stepping (strong stability restrictions):

$$\frac{u_i^{n+1} - u_i^n}{\delta t} = 8\frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{(\delta x)^2}$$

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Finite Difference Method for Heat Equation

- Simple method to derive and implement
- Hardest part for implicit schemes is solution of resulting linear system of equations
- Explicit schemes typically have stability restrictions or can always be unstable
- Convergence rates tend not to be great – to get an accurate solution, a large number of grid points are needed
- Difficult to use for complicated geometries
- Tends to scale well since communication requirements are low

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Finite Volume Method for Heat Equation

- $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$ or $\frac{\partial u}{\partial x} = v$ and $\frac{\partial u}{\partial t} = \frac{\partial v}{\partial x}$
- Consider a cell averaged integral, then use implicit midpoint rule

$$\int_{x_i}^{x_{i+1}} u_x \mathrm{d}x = \int_{x_i}^{x_{i+1}} v \mathrm{d}x$$
$$u_{i+1} - u_i = \int_{x_i}^{x_{i+1}} v \mathrm{d}x$$
$$\frac{u_{i+1}^{n+1} + u_{i+1}^{n} - u_i^{n+1} - u_i^n}{2} = \frac{v_{i+1}^{n+1} + v_{i+1}^{n} + v_i^{n+1} + v_i^n}{4} \delta x$$

$$\int_{x_i}^{x_{i+1}} u_t \mathrm{d}x = \int_{x_i}^{x_{i+1}} v_x \mathrm{d}x$$
$$\int_{x_i}^{x_{i+1}} u_t \mathrm{d}x = v_{i+1} - v_i$$
$$\frac{u_{i+1}^{n+1} + u_{i}^{n+1} - u_{i+1}^{n} - u_i^n}{2\delta t} \delta x = \frac{v_{i+1}^{n+1} + v_{i+1}^{n} - v_i^{n+1} - v_i^n}{2}$$

- Several ways of approximating the integrals. The one above is a little unusual, most finite volume schemes use left sided or right sided approximations.

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Finite Volume Method for Heat Equation

- For implicit schemes, hardest part is solving the system of equations that results
- Explicit schemes parallelize very well, however a large number of grid points are usually needed to get accurate results
- Automated construction of simple finite volume schemes is possible, making them popular in packages
- No convergence theory for high order finite volume schemes
- Tricky to do complicated geometries accurately

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Finite Element Method for Heat Equation

- $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$
- Assume $u(x,t) \approx \tilde{u}(x,t) = \sum_i \phi_i(x) T_i(t)$, where $\phi_i(x)$ is zero for $x > x_{i+1}$ and $x < x_{i-1}$.
- A simple choice is ψ is the triangle hat function, $x_{i+1} - x$ for $x \in [x_i, x_{i+1}]$ and $x - x_{i-1}$ for $x \in [x_{i-1}, x_i]$
- Now need to find $T_i(t)$, which will be evaluated using finite differences.

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Finite Element Method for Heat Equation

- Consider multiplying the heat equation by a polynomial, ϕ_j that is only non zero over a few grid points, then integrate by parts,

$$\int_{x_{i-1}}^{x_{i+1}} \frac{\partial \tilde{u}}{\partial t} \phi_i \mathrm{d}x = \int_{x_{i-1}}^{x_{i+1}} \frac{\partial^2 \tilde{u}}{\partial x^2} \phi_i \mathrm{d}x$$

$$\int_{x_{i-1}}^{x_{i+1}} \frac{\partial \tilde{u}}{\partial t} \phi_i \mathrm{d}x = - \int_{x_{i-1}}^{x_{i+1}} \frac{\partial \tilde{u}}{\partial x} \frac{\partial \phi_i}{\partial x} \mathrm{d}x$$

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Finite Element Method for Heat Equation

- Then use implicit midpoint rule

$$\int_{x_{i-1}}^{x_{i+1}} \frac{\partial \tilde{u}}{\partial t}\phi_i \mathrm{d}x = -\int_{x_{i-1}}^{x_{i+1}} \frac{\partial \tilde{u}}{\partial x}\frac{\partial \phi_i}{\partial x}\mathrm{d}x$$

$$\int_{x_{i-1}}^{x_{i+1}} \frac{\tilde{u}^{n+1}-\tilde{u}^{n}}{\delta t}\phi_i \mathrm{d}x = -\int_{x_{i-1}}^{x_{i+1}} \frac{1}{2}\left(\frac{\partial \tilde{u}^{n}}{\partial x}+\frac{\partial \tilde{u}^{n+1}}{\partial x}\right)\frac{\partial \phi_i}{\partial x}\mathrm{d}x$$

$$\frac{1}{2}\left[\frac{\tilde{u}^{n+1}_{i+1}-\tilde{u}^{n}_{i+1}}{\delta t}\phi_i\big|_{x_{i+1}} + \frac{\tilde{u}^{n+1}_{i-1}-\tilde{u}^{n}_{i-1}}{\delta t}\phi_i\big|_{x_{i-1}}\right]2\delta x$$

$$= -\frac{1}{2}\left[\frac{1}{2}\left(\frac{\partial \tilde{u}^{n+1}}{\partial x}\bigg|_{x_{i+1}} + \frac{\partial \tilde{u}^{n}}{\partial x}\bigg|_{x_{i+1}}\right)\frac{\partial \phi_i}{\partial x}\bigg|_{x_{i+1}}\right]2\delta x$$

$$-\frac{1}{2}\left[\frac{1}{2}\left(\frac{\partial \tilde{u}^{n+1}}{\partial x}\bigg|_{x_{i-1}} + \frac{\partial \tilde{u}^{n}}{\partial x}\bigg|_{x_{i-1}}\right)\frac{\partial \phi_i}{\partial x}\bigg|_{x_{i-1}}\right]2\delta x$$

- Since ϕ are known before hand, can re-write this as matrix vector products, so need to solve a linear system at each time step.

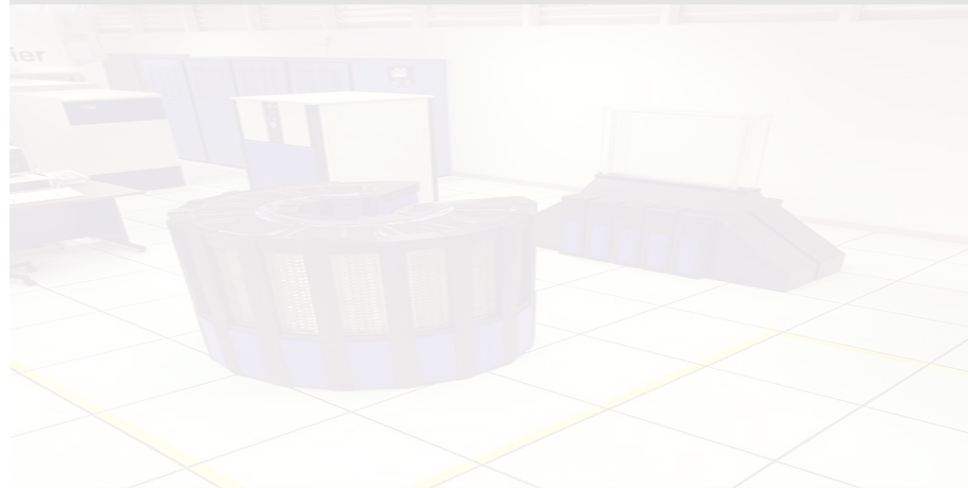
By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Finite Element Method for Heat Equation

- Several other ways of approximating the integrals, can extend to multiple dimensions.
- Weak formulation allows for solution of equations where second derivative is not naturally defined
- Large mathematical community developing convergence theory for these methods
- Well suited to complicated geometries
- Rather difficult to implement compared to other schemes because of integrals that need to be computed
- Used in many codes, but typically codes are hand written to obtain high efficiency
- For implicit time discretizations, solving the linear system of equations that results can be most time consuming part

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Fourier Spectral Method for Heat Equation



By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Fourier Series: Separation of Variables 1

$$
\begin{aligned}
\frac{dy}{dt} &= y \\
\frac{dy}{y} &= dt \\
\int \frac{dy}{y} &= \int dt \\
\ln y + a &= t + b \\
e^{\ln y + a} &= e^{t+b} \\
e^{\ln y} e^{a} &= e^{t} e^{b} \\
y &= \frac{e^{b}}{e^{a}} e^{t} \\
y(t) &= ce^{t}
\end{aligned}
$$

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Fourier Series: Separation of Variables 2

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

- Suppose $u = X(x)T(t)$

$$\frac{\frac{\mathrm{d}T}{\mathrm{d}t}(t)}{T(t)} = \frac{\frac{\mathrm{d}^2 X}{\mathrm{d}x^2}(x)}{X(x)} = -C,$$

- Solving each of these separately and then using linearity we get a general solution

$$\sum_{n=0}^{\infty} \alpha_n \exp(-C_n t)\sin(\sqrt{C_n}x) + \beta_n \exp(-C_n t)\cos(\sqrt{C_n}x)$$

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Fourier Series: Separation of Variables 3

- How do we find a particular solution?
- Suppose $u(x, t = 0) = f(x)$
- Suppose $u(0, t) = u(2\pi, t)$ and $u_x(0, t) = u_x(2\pi, t)$ then recall
-

$$\int_0^{2\pi} \sin(nx)\sin(mx) = \begin{cases} \pi & m = n \\ 0 & m \neq n \end{cases},$$

$$\int_0^{2\pi} \cos(nx)\cos(mx) = \begin{cases} \pi & m = n \\ 0 & m \neq n \end{cases},$$

$$\int_0^{2\pi} \cos(nx)\sin(mx) = 0.$$

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Fourier Series: Separation of Variables 4

- So if

$$f(x) = \sum_{n=0}^{\infty} \alpha_n \sin(nx) + \beta_n \cos(nx).$$

- then

$$\alpha_n = \frac{\int_0^{2\pi} f(x) \sin(nx) \mathrm{d}x}{\int_0^{2\pi} \sin^2(nx) \mathrm{d}x}$$

$$\beta_n = \frac{\int_0^{2\pi} f(x) \cos(nx) \mathrm{d}x}{\int_0^{2\pi} \cos^2(nx) \mathrm{d}x}.$$

- and

$$u(x,t) = \sum_{n=0}^{\infty} \exp(-n^2 t) [\alpha_n \sin(nx) + \beta_n \cos(nx)]$$

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

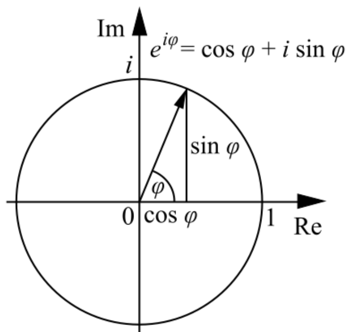
Fourier Series: Separation of Variables 5

- The Fast Fourier Transform allows one to find good approximations to α_n and β_n when the solution is found at a finite number of evenly spaced grid points
- By rescaling, can consider intervals other than $[0, 2\pi)$
- Fourier transform also works on infinite intervals, but require function to decay to the same constant value at $\pm\infty$

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Complex Fourier Series

- By using Euler's formula, one can get a simpler expression for a Fourier series where sine and cosine are combined
- $u = \sum_{n=-\infty}^{\infty} \gamma_n \exp(inx) \quad x \in [0, 2\pi)$



Source: `http://en.wikipedia.org/wiki/File:Euler%27s_formula.svg`

A Computational Algorithm for Computing An Approximate Fourier Transform 1

- Analytic method of computing Fourier transform can be tedious
- Can use quadrature to numerically evaluate Fourier transforms – $O(n^2)$ operations
- Gauss and then Cooley and Tukey found $O(n \log n)$ algorithm
- Key observation is to use factorization and recursion
- Modern computers use variants of this idea that are more suitable for computer hardware where moving data is more expensive than floating point operations

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

A Computational Algorithm for Computing An Approximate Fourier Transform 2

Example pseudo code to compute a radix 2 out of place DFT where x has length that is a power of 2

1: **procedure** $X_{0,\dots,N-1}$(ditfft2(x,N,s))
2: *DFT of* $(x_0, x_s, x_2 s, \dots, x_{(N-1)s})$
3: **if** N=1 **then**
4: *trivial size-1 DFT of base case*
5: $X_0 \leftarrow X_0$
6: **else**
7: *DFT of* $(x_0, x_{2s}, x_{4s}, \dots)$
8: $X_{0,\dots,N/2-1} \leftarrow ditfft2(x, N/2, 2s)$
9: *DFT of* $(x_s, x_{s+2s}, x_{s+4s}, \dots)$
10: $X_{N/2,\dots,N-1} \leftarrow ditfft2(x+s, N/2, 2s)$
11: *Combine DFTs of two halves into full DFT*
12: **for** $k = 0 \to N/2-1$ **do**
13: $t \leftarrow X_k$
14: $X_k \leftarrow t + \exp(-2\pi ik/N) X_{k+N/2}$
15: $X_{k+N/2} \leftarrow t - \exp(-2\pi ik/N) X_{k+N/2}$
16: **end for**
17: **end if**
18: **end procedure**

Sources: http://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey_FFT_algorithm, http://cnx.org/content/m16336/latest/

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

The Heat Equation: Finding Derivatives and Timestepping

- Let

$$u(x) = \sum_k \hat{u}_k \exp(ikx)$$

- then

$$\frac{\mathrm{d}^\nu u}{\mathrm{d}x^\nu} = \sum (ik)^\nu \hat{u}_k \exp(ikx).$$

- Consider $u_t = u_{xx}$, which is approximated by

$$\begin{aligned}
\frac{\partial \hat{u}_k}{\partial t} &= (ik)^2 \hat{u}_k \\
\frac{\hat{u}_k^{n+1} - \hat{u}_k^n}{\delta t} &= (ik)^2 \hat{u}_k^{n+1} \\
\hat{u}_k^{n+1}[1 - \delta t(ik)^2] &= \hat{u}_k^n \\
\hat{u}_k^{n+1} &= \frac{\hat{u}_k^n}{[1 - \delta t(ik)^2]}.
\end{aligned}$$

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

The Heat Equation: Finding Derivatives and Timestepping

- Python demonstration

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

The Allen Cahn Equation: Implicit-Explicit Method

- Consider $u_t = \varepsilon u_{xx} + u - u^3$, which is approximated by *backward Euler* for the linear terms and *forward Euler* for the nonlinear terms

$$
\begin{aligned}
\frac{\partial \hat{u}}{\partial t} &= \varepsilon (ik)^2 \hat{u} + \hat{u} - \widehat{u^3} \\
\frac{\hat{u}^{n+1} - \hat{u}^n}{\delta t} &= \varepsilon (ik)^2 \widehat{u^{n+1}} + \widehat{u^n} - (\widehat{u^n})^3
\end{aligned}
$$

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

The Allen Cahn Equation: Implicit-Explicit Method

- Python demonstration

Fast Fourier Transform Libraries

- 2DECOMP&FFT http://www.2decomp.org/
- P3DFFT http://p3dfft.net/
- PFFT https://www-user.tu-chemnitz.de/~mpip/software.php#pfft
- PARRAY/PKUFFT https://code.google.com/p/parray-programming/
- AccFFT http://accfft.org
- mpiFFT4py https://github.com/spectralDNS/mpiFFT4py
- FFTE http://www.ffte.jp/
- Intel MKL cluster FFT https://software.intel.com/en-us/node/521991
- FFTW http://fftw.org/

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

The Real Cubic Klein-Gordon Equation

- $$u_{tt} - \Delta u + u = |u|^2 u$$
- $$E(u, u_t) = \int \frac{1}{2}|u_t|^2 + \frac{1}{2}|u|^2 + \frac{1}{2}|\nabla u|^2 - \frac{1}{4}|u|^4 \, \mathrm{d}\boldsymbol{x}$$

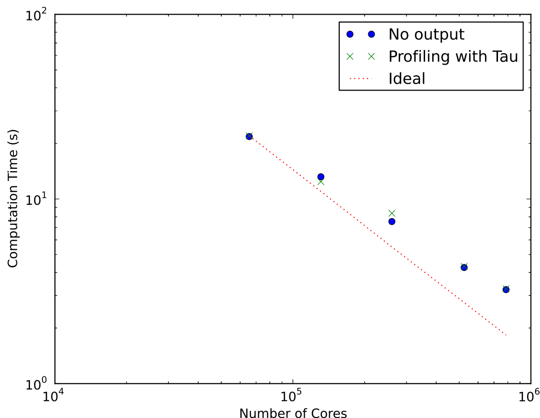
By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Videos by Brian Leu, Albert Liu, Michael Quell and Parth Sheth

- http://www-personal.umich.edu/~alberliu/
- http://www-personal.umich.edu/~brianleu/
- http://www-personal.umich.edu/~pssheth/
- https://www.youtube.com/watch?v=nTA3zfgNfQg

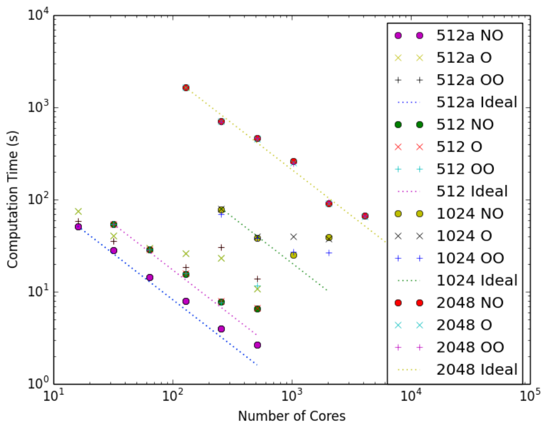
By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Scaling study on Mira



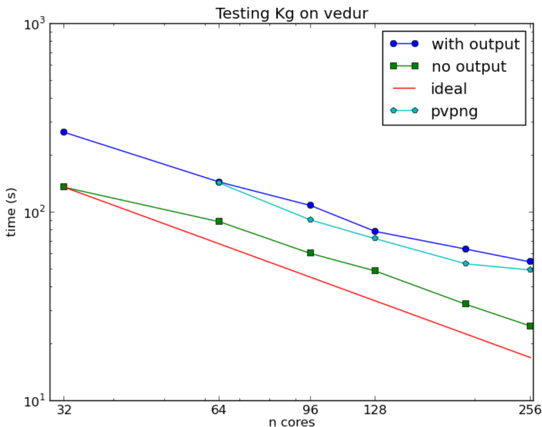
- Strong scaling on Mira for a 4096^3 discretization by Brian Leu, Albert Liu, and Parth Sheth

Scaling study on Stampede



- Strong scaling on Stampede by Jerome Vienne

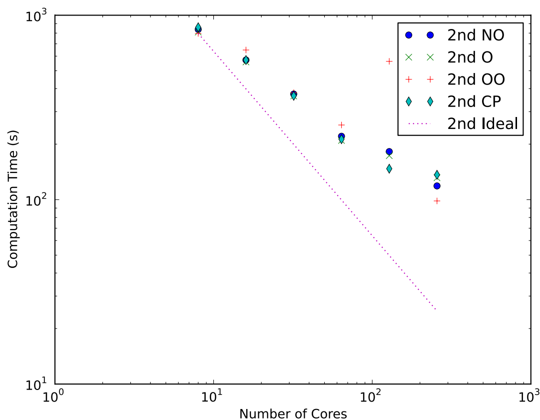
Scaling study on Vedur



- Strong scaling on Vedur by Oleg Batrašev

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Scaling study on Neser



- Strong scaling on Neser by Samar Aseeri

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Numerical Scheme

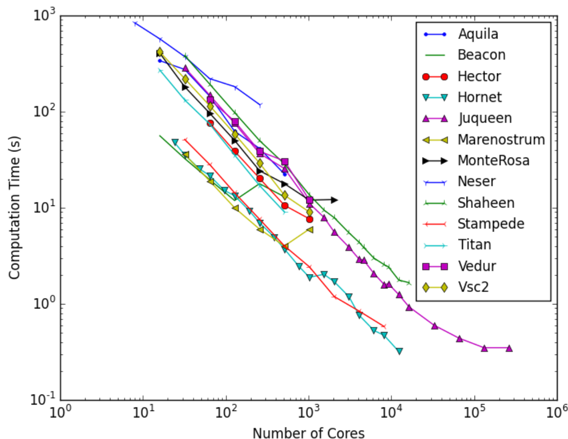
-

$$\frac{u^{n+1}-2u^n+u^{n-1}}{\delta t^2}-\Delta\frac{u^{n+1}+2u^n+u^{n-1}}{4}$$
$$+\frac{u^{n+1}+2u^n+u^{n-1}}{4}=\left|u^n\right|^2u^n$$

- $u^n \approx u(n\delta t, x, y, z)$
- Time stepping takes place in Fourier space
- Solution of linear system of equations is easy in Fourier space, though can also be done by iterative methods in real space
- Two FFTs per time step

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

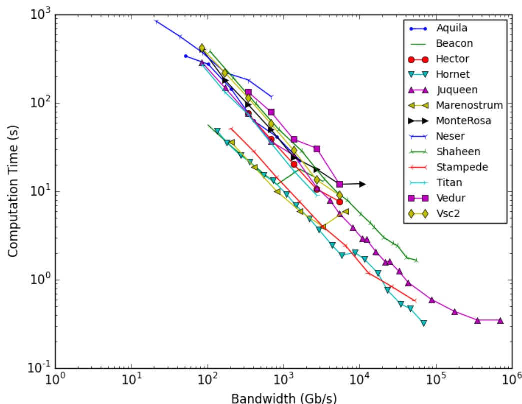
Scaling with Cores



- Scaling results showing computation time for 30 time steps as a function of the number of processor cores. A discretization of 512^3 points was used.

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Scaling with Cores



- Scaling results showing computation time for 30 time steps as a function of total on chip bandwidth defined as the maximum theoretical bandwidth from RAM on a node multiplied by the number of nodes used. A discretization of 512^3 points was used.

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

A Runtime Estimation Model

- d_1, d_2, d_3 system and implementation dependent constants
- N number of grid points in each dimension, assumed to be the same in all three dimensions
- L_n minimum network latency, B_c average bandwidth to a core from RAM
- p number of processes
- Assume a hypercube network - speed optimal for FFT
- $$\frac{d_1 \times N^3 + d_2 \times [N \log(N)]^3}{B_c \times p} + 2L_n + d_3 \log(p)$$

A Ranking

Rank	Machine Name	Time (s)	Cores used	Manufacturer and Model	Node Type	Total Cores
1	Hornet	0.319	12,288	Cray XC40	2x12 core Intel Xeon 2.5 GHz E5-2680v3	94,656
2	Juqueen	0.350	262,144	IBM Blue Gene/Q	16 core 1.6 GHz Power PC A2	458,752
3	Stampede	0.581	8,162	Dell Power Edge	2x8 core Intel Xeon 2.7 GHz E5-2680	462,462
4	Shaheen	1.66	16,384	IBM Blue Gene/P	4 core 0.85 GHz PowerPC 450	65,536
5	MareNostrum III	4.00	64	IBM DataPlex	2x8 core Intel Xeon 2.6 GHz E5-2670	48,384
6	Hector	7.66	1024	Cray XE6	2x16 core AMD Opteron 2.3 GHz 6276 16C	90,112
7	VSC2	9.03	1024	Megware	2x8 core AMD Opteron 2.2 GHz 6132HE	21,024
8	Beacon	9.13	256	Appro	2x8 core Intel Xeon 2.6 GHz E5-2670	768
9	Monte Rosa	11.9	1,024	Cray XE6	2x16 core AMD Opteron 2.1 GHz 6272	47,872
10	Titan	17.0	256	Cray XK7	16 core AMD Opteron 2.2 GHz 6274	299,008
11	Vedur	18.6	1,024	HP ProLiant DL165 G7	2x16 core AMD Opteron 2.3 GHz 6276	2,560
12	Aquila	22.4	256	ClusterVision	2x4 core Intel Xeon 2.8 GHz E5462	800
13	Neser	118.7	128	IBM System X3550	2x4 core Intel Xeon 2.5 GHz E5420	1,024

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

A Ranking

Rank	Machine Name	Time (s)	Total Cores	Interconnect	1D FFT Library	Chip Bandwidth Gb/s	Theoretical Peak TFLOP/s
1	Hornet	0.319	94,656	Cray Aries	FFTW 3	68	3,784
2	Juqueen	0.350	458,752	IBM 5D torus	ESSL	42.6	5,872
3	Stampede	0.581	462,462	FDR infiniband	Intel MKL	51.2	2,210
4	Shaheen	1.66	65,536	IBM 3D torus	ESSL	13.6	222.8
5	MareNostrum III	4.00	48,384	FDR10 infiniband	Intel MKL	51.2	1,017
6	Hector	7.66	90,112	Cray Gemini	ACML	85	829.0
7	VSC2	9.03	21,024	QDR infiniband	FFTW 3	42.8	185.0
8	Beacon	9.13	768	FDR infiniband	Intel MKL	51.2	16.0
9	Monte Rosa	11.9	47,872	Cray Gemini	ACML	85	402.1
10	Titan	17.0	299,008	Cray Gemini	ACML	85	2,631
11	Vedur	18.6	2,560	QDR infiniband	FFTW 3	85	236
12	Aquila	22.4	800	DDR infiniband	FFTW 3	12.8	8.96
13	Neser	118.7	1,024	Gigabit ethernet	FFTW 3	10.7	10.2

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Current work

- Extending to other Fourier transform libraries
- Trying other linear solvers
- Adding visualization
- Trying other time stepping algorithms
- Model problems from other areas

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Questions

- Is a ranking based on one metric useful?
- What would encourage people to optimize algorithms on new computer architectures?
- How should one benchmark input and output?
- If run a fixed non-optimal benchmark, can one use this to produce a predictive model for other programs?
- If optimize a benchmark, do good results encourage code development to make use of new hardware?
- What benchmark results would enable you to build a small cluster for your workflow?
- How develop realistic communication network performance models?
- Would an HPC `openbenchmarking.org` be useful?
- Would similar efforts for IO `http://www.vi4io.org/` be useful?

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Acknowledgements

The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by:

- The Beacon Project at the University of Tennessee;
- The UK's national high-performance computing service;
- The Barcelona Supercomputing Center - Centro Nacional de Supercomputación;
- The Swiss National Supercomputing Centre (CSCS);
- The Texas Advanced Computing Center (TACC) at The University of Texas at Austin;
- The KAUST Supercomputer Laboratory at King Abdullah University of Science and Technology (KAUST);
- The Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory;
- The Aquila HPC service at the University of Bath;
- The Vienna Scientific Cluster (VSC);
- The PRACE research infrastructure resources in Germany at HLRS and FZ Jülich;
- The High Performance Computing Center of the University of Tartu;
- Kraken at the National Institute for Computational Science;
- Trestles at the San Diego Supercomputing Center;
- the University of Michigan High Performance Computing Service FLUX;
- Mira at the Argonne Leadership Computing Facility at Argonne National Laboratory;

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding bodies or the service providers.

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

Acknowledgments

- D. Acevedo-Feliz, S. Andersson, W. Auzinger, A. Bauer, D. DeMarle, L. Dorogin, D. Ketcheson, D. Keyes, R. Krasny, D. Pekurovsky, M. Pippig, P. Rigge, S. Shende, M. Srinivasan, E. Vainikko, M. Winkel, B. Wylie, H. Yi and R. Yokota
- S. Aseeri, O. Batrašev, M. Icardi, B.Leu, A. Liu, N. Li, E. Müller, B. Palen, M. Quell, H. Servat, P. Sheth, R. Speck, M. Van Moer, J. Vienne,
- The Blue Waters Undergraduate Petascale Education Program administered by the Shodor foundation
- The Division of Literature, Sciences and Arts at the University of Michigan

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.

References

- To follow, will be posted separately

By NASA - http://gimp-savvy.com/cgi-bin/img.cgi?ailswE7kkmL1216740, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1328034.