Delivering performance in scientific simulations: present and future role of GPUs in supercomputers

Thomas C. Schulthess
What exactly is good performance?
What we really should care about is …

Energy & Time
But what about FLOPs and GFlops/Watt

Is Top500 / Green500 meaningless?

Metric for time to solution in High-Performance LINPACK benchmark:

1. high arithmetic density: \[ \frac{\text{# of flop}}{\text{# of load-stores}} \propto O(N) \]
2. total work measured is number of retired floating point operations (\( \text{tot flop} \)), can be easily computed
3. normalized time to solution: \[ \frac{\Delta t_c}{\text{tot flop}} \]
   or performance \[ \frac{\text{tot flop}}{\Delta t_c} \left[ \frac{\text{flop}}{\text{sec.}} = \text{flops} \right] \]

... and a metric for energy to solution of HPL:

1. normalized energy to solution \( E \) by simple measure of work \[ \frac{E}{\text{tot flop}} \]
2. minimizing energy to solution is equivalent to maximizing \[ \frac{\text{tot flop}}{E} \left[ \frac{\text{flop}}{\text{Joule}} \right] \]
3. ... and of course \[ \left[ \frac{\text{flop}}{\text{Joule}} \right] = \left[ \frac{\text{flop}}{\text{sec.}} \right] \left[ \frac{\text{sec.}}{\text{Watt}} \right] \]
So High-Performance Linpack, Top500, and Green500 are meaningful after all

- High-Performance Linpack (HPL) or, more generally, dense linear algebra is an algorithmic motif used in many applications (e.g. quantum simulations)

- Minimising time to solution in HPL (dense linear algebra motifs) is equivalent to maximising floating point per seconds (i.e. ranking on Top500)

- Minimising energy to solution is equivalent to maximising \([\text{flop/Joule}]\) or \([\text{Gigaflops/Watt}]\) (i.e. ranking on Green500)

Top500 / Green500 are meaningful for certain applications
COSO: limited-area climate modelling application

- Consortium for Small-Scale MOdeling

- Limited-area climate model (http://www.cosmo-model.org)

- Used by 7 weather services as well as ~50 universities & research institutes
COSMO in production at Meteo Swiss

COSMO-7
3x per day 72h forecast
6.6 km lateral grid, 60 layers

ECMWF
2x per day
16 km lateral grid, 91 layers

COSMO-2
8x per day 24h forecast
2.2 km lateral grid, 60 layers

Some of the products generated with COSMO-2
- Daily weather forecast
- Forecasting for Swiss air traffic control (Sky Gide)
- Safety management in events of nuclear incidents
Performance profile of original COSMO-2 model

Runtime based 2 km production model of MeteoSwiss

% Code Lines (F90) % Runtime

- Assimilation: 83,271; 37%
- Dynamics: 235,7; 22%
- Physics: 64,98; 6%
- I/O: 102,9; 10%
- Structure: 12,86; 1%
- Diagnosis: 41,548; 18%
- Parallelization: 43,066; 19%
- Other: 11,031; 5%
- Other: 11,079; 5%
- Other: 25,300; 11%
- Other: 12,094; 5%
Typical code snippets

Physics

```fortran
do j = 1, niter
   do i = 1, nwork
      c(i) = a(i) + b(i) + sin(b(i)) * log(a(i))
   end do
end do
```

3 memory accesses 136 FLOPs ➔ compute bound

Dynamics

```fortran
do j = 1, niter
   do i = 1, nwork
      c(i) = a(i) + b(i) * (a(i+1) - 2.0d0*a(i) + a(i-1))
   end do
end do
```

3 memory accesses 5 FLOPs ➔ memory bound
Running the snippets on a Cray XK6 (early 2012)

### Compute bound (physics) problem

<table>
<thead>
<tr>
<th>Machine</th>
<th>Interlagos</th>
<th>Fermi (2090)</th>
<th>GPU+transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time</strong></td>
<td>1.31 s</td>
<td>0.17 s</td>
<td>1.9 s</td>
</tr>
<tr>
<td><strong>Speedup</strong></td>
<td>1.0 (REF)</td>
<td>7.6</td>
<td>0.7</td>
</tr>
</tbody>
</table>

### Memory bound (dynamics) problem

<table>
<thead>
<tr>
<th>Machine</th>
<th>Interlagos</th>
<th>Fermi (2090)</th>
<th>GPU+transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time</strong></td>
<td>0.16 s</td>
<td>0.038 s</td>
<td>1.7 s</td>
</tr>
<tr>
<td><strong>Speedup</strong></td>
<td>1.0 (REF)</td>
<td>4.2</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Pay attention to data management (leave data on GPU)
Performance profile of original COSMO-2 model

Runtime based 2 km production model of MeteoSwiss

% Code Lines (F90) % Runtime

Original code (with OpenACC for GPU) Rewrite in C++ (with CUDA backend for GPU)
Insights into model, methods, and algorithms used in simulations with COSMO

- Solve PDE on structured grid where variables are velocity, temperature, pressure, humidity, etc.
- Explicit solve in horizontally (I,J) using finite differences
- Implicit solve in vertical direction (K), i.e. tridiagonal solve in every column

Implicit solve over columns allows use of longer time steps (now determined by 2km lateral grid rather than 60m short spacing)
Stencil computation

Stencils update array elements with a fixed pattern

\[
\text{lap}(i,j,k) = -4.0 \times \text{data}(i,j,k) + \text{data}(i+1,j,k) + \text{data}(i-1,j,k) + \text{data}(i,j+1,k) + \text{data}(i,j-1,k);
\]

e.g. Laplacian: 5 flops per 6 load/stores \( \sim 0.1 \) flops/Byte

Kepler K20x GPU has 1310 Gflops / 200 GB/sec. \( \sim 6.6 \) flops/Byte

Stencil computations are memory bandwidth bound
To make matters more complicated

Tridiagonal solve in vertical
  ➜ loop carried dependencies in K
  ➜ focus on K access

Horizontal stencil computations
  ➜ focus on IJ access
  ➜ no loop carried dependencies

The two main motifs used in COSMO have different data access patterns – optimal storage order may depend on architecture
Dynamics in COSMO

\[
\frac{\partial u}{\partial t} = - \left\{ \frac{1}{a \cos \varphi} \frac{\partial E_h}{\partial \lambda} - v V_a \right\} - \frac{\xi}{\rho a \cos \varphi} \left( \frac{\partial p'}{\partial \varphi} - \frac{1}{\sqrt{\gamma}} \frac{\partial p_0}{\partial \varphi} \right) + M_u
\]

velocities

\[
\frac{\partial v}{\partial t} = - \left\{ \frac{1}{a \varphi} \frac{\partial E_h}{\partial \varphi} + u V_a \right\} - \frac{\xi}{\rho a} \left( \frac{\partial p'}{\partial \varphi} - \frac{1}{\sqrt{\gamma}} \frac{\partial p_0}{\partial \varphi} \right) + M_v
\]

\[
\frac{\partial w}{\partial t} = - \left\{ \frac{1}{a \cos \varphi} \left( u \frac{\partial w}{\partial \lambda} + v \cos \varphi \frac{\partial w}{\partial \varphi} \right) \right\} - \frac{\xi}{\rho} \frac{\partial p'}{\partial \varphi} + g \frac{\rho_0}{\sqrt{\gamma}} \frac{\partial p'}{\partial \zeta} + M_u + g \frac{\rho_0}{\rho} \left\{ \frac{T - T_0}{T} - \frac{T_0 p'}{T p_0} + \left( \frac{R_v}{R_d} - 1 \right) q^v - q - q^f \right\}
\]

pressures

\[
\frac{\partial T}{\partial t} = - \left\{ \frac{1}{a \cos \varphi} \left( u \frac{\partial T}{\partial \lambda} + v \cos \varphi \frac{\partial T}{\partial \varphi} \right) \right\} - \frac{\xi}{\rho c_p} \frac{\partial p'}{\partial \varphi} + g \frac{\rho_0}{c_p} w - \frac{c_p d}{c_v d} p D
\]

temperature

\[
\frac{\partial q^w}{\partial t} = - \left\{ \frac{1}{a \cos \varphi} \left( u \frac{\partial q^w}{\partial \lambda} + v \cos \varphi \frac{\partial q^w}{\partial \varphi} \right) \right\} - \frac{\xi}{\rho} \frac{\partial q^w}{\partial \varphi} + (S^l + S^f) + M_{q^w}
\]

water

\[
\frac{\partial q^{l,f}}{\partial t} = - \left\{ \frac{1}{a \cos \varphi} \left( u \frac{\partial q^{l,f}}{\partial \lambda} + v \cos \varphi \frac{\partial q^{l,f}}{\partial \varphi} \right) \right\} - \frac{\xi}{\rho} \frac{\partial q^{l,f}}{\partial \varphi} + S^{l,f} + M_{q^{l,f}}
\]

turbulence

\[
\frac{\partial e_t}{\partial t} = - \left\{ \frac{1}{a \cos \varphi} \left( u \frac{\partial e_t}{\partial \lambda} + v \cos \varphi \frac{\partial e_t}{\partial \varphi} \right) \right\} - \frac{\xi}{\rho} \frac{\partial e_t}{\partial \varphi} + K_m \frac{g \rho_0}{\sqrt{\gamma}} \left\{ \left( \frac{\partial u}{\partial \zeta} \right)^2 + \left( \frac{\partial v}{\partial \zeta} \right)^2 \right\} + \frac{g}{\rho e} \frac{\theta}{\alpha M l}^3 - \frac{\sqrt{2 e_t^{3/2}}}{\alpha M l} + M_{e_t}
\]

Timestep

- implicit (sparse)
- explicit (RK3)
- implicit (sparse solver)
- explicit (leapfrog)

1x

- physics et al. tendencies
- horizontal adv.
- vertical adv.
- 3x
- fast wave solver
- 10x

water adv.

CSCS

Centre for Scientific Computing
Swiss National Supercomputing Centre

ETH Zürich

Thursday, March 26, 2014, GTC’14, San Jose  T. Schulthess  15
Implement dynamics in terms of a stencil library

- Generic library using C++ and template meta-programming
  - currently support 3D structured grid
  - parallel over horizontal IJ-plane (sequential in K for tridiagonal solve)
  - multi-node support with halo exchange (Generic Communication Library)
- Abstract hardware platform (library runs optimally on CPUs and GPUs)
  - adapt loop order and storage layout to hardware platform
  - leverage software caching
- Hide complex optimisation from users / programmer
- Since source code compiles to multiple platforms
  - current backends support X86 and CUDA

<table>
<thead>
<tr>
<th>Storage Order (Fortran notation)</th>
<th>X86 CPU</th>
<th>Tesla GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>KIJ</td>
<td>OpenMP</td>
<td>CUDA</td>
</tr>
<tr>
<td>IJK</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Software architecture overview of current and new COSMO production code

**main (current)**
- **physics**
- **dynamics**
- **MPI**

**system**

**main (new)**
- **dynamics**
- **physics with OpenMP / OpenACC**
- **stencil library**
- **X86**
- **GPU**
- **boundary conditions**
- **GCL**
- **MPI**

**system**

**Basic Libraries (incl. BLAS, LAPACK, FFT, ...)**

**Domain Specific Libraries & Tools**

**Application code / tool**

**Prototyping code / interactive data analysis**
COSMO dynamical core running on CPU or GPU

Mesh dimensions vs. wall time/time step (s)

- Sandy Bridge E5–2670 Socket
- M2090
- K20X

Current production:
- High throughput running on fewer nodes on GPU

High performance running on more nodes on CPU
Three ways to look at hybrid CPU-GPU blades

Standard: GPU is an accelerator

“Distributed latency optimized cores”

“Distributed throughput optimized cores”

Depending on workload, COSMO runs in either of these two modes:

> domain decompose data over DDR3 or GDDR5 memory
> move the computation close to data
> exchange halos between CPUs or GPUs (with G-2-G communications)
Speedup of the full COSMO-2 production problem (apples to apples with 33h forecast of Meteo Swiss)

Current production code

New HP2C funded code

all runs performed on same number of nodes
Energy to solution (kWh / ensemble member)

Cray XE6 (Nov. 2011) 1.75x
Cray XK7 (Nov. 2012) 1.41x
Cray XC30 (Nov. 2012) 1.75x
Cray XC30 hybrid (GPU) (Nov. 2013) 1.49x

Current production code

New HP2C funded code

6.0
4.5
3.0
1.5

2.51x
1.49x
2.64x
3.93x
6.89x
Domain science (incl. applied mathematics)

\[
\begin{align*}
\frac{\partial u}{\partial t} &= -\left\{ \frac{1}{a \cos \varphi} \frac{\partial E_h}{\partial x} - v \nu \right\} - \frac{\partial}{\partial c} \left\{ \frac{\partial u}{\partial c} \right\} + \frac{\partial}{\partial x} \left( \frac{\partial u}{\partial x} \right) + M_u \\
\frac{\partial v}{\partial t} &= -\left\{ \frac{1}{a \cos \varphi} \frac{\partial E_h}{\partial y} + u \nu \right\} - \frac{\partial}{\partial c} \left\{ \frac{\partial v}{\partial c} \right\} + \frac{\partial}{\partial y} \left( \frac{\partial v}{\partial y} \right) + M_v \\
\frac{\partial w}{\partial t} &= -\left\{ \frac{1}{a \cos \varphi} \frac{\partial E_h}{\partial z} + v \nu \right\} - \frac{\partial}{\partial c} \left\{ \frac{\partial w}{\partial c} \right\} + \frac{\partial}{\partial z} \left( \frac{\partial w}{\partial z} \right) + M_w \\
\frac{\partial \rho}{\partial t} &= -\left\{ \frac{1}{a \cos \varphi} \frac{\partial \rho}{\partial x} + v \nu \right\} - \frac{\partial}{\partial c} \left\{ \frac{\partial \rho}{\partial c} \right\} + \frac{\partial}{\partial x} \left( \frac{\partial \rho}{\partial x} \right) + M_\rho \\
\frac{\partial T}{\partial t} &= -\left\{ \frac{1}{a \cos \varphi} \frac{\partial T}{\partial x} + v \nu \right\} - \frac{\partial}{\partial c} \left\{ \frac{\partial T}{\partial c} \right\} + \frac{\partial}{\partial x} \left( \frac{\partial T}{\partial x} \right) + M_T \\
\frac{\partial \xi}{\partial t} &= -\left\{ \frac{1}{a \cos \varphi} \frac{\partial \xi}{\partial x} + v \nu \right\} - \frac{\partial}{\partial c} \left\{ \frac{\partial \xi}{\partial c} \right\} + \frac{\partial}{\partial x} \left( \frac{\partial \xi}{\partial x} \right) + M_\xi \\
\frac{\partial \eta}{\partial t} &= -\left\{ \frac{1}{a \cos \varphi} \frac{\partial \eta}{\partial y} + v \nu \right\} - \frac{\partial}{\partial c} \left\{ \frac{\partial \eta}{\partial c} \right\} + \frac{\partial}{\partial y} \left( \frac{\partial \eta}{\partial y} \right) + M_\eta \\
\frac{\partial \zeta}{\partial t} &= -\left\{ \frac{1}{a \cos \varphi} \frac{\partial \zeta}{\partial z} + v \nu \right\} - \frac{\partial}{\partial c} \left\{ \frac{\partial \zeta}{\partial c} \right\} + \frac{\partial}{\partial z} \left( \frac{\partial \zeta}{\partial z} \right) + M_\zeta \\
\frac{\partial \phi}{\partial t} &= -\left\{ \frac{1}{a \cos \varphi} \frac{\partial \phi}{\partial x} + v \nu \right\} - \frac{\partial}{\partial c} \left\{ \frac{\partial \phi}{\partial c} \right\} + \frac{\partial}{\partial x} \left( \frac{\partial \phi}{\partial x} \right) + M_\phi \\
\frac{\partial \psi}{\partial t} &= -\left\{ \frac{1}{a \cos \varphi} \frac{\partial \psi}{\partial y} + v \nu \right\} - \frac{\partial}{\partial c} \left\{ \frac{\partial \psi}{\partial c} \right\} + \frac{\partial}{\partial y} \left( \frac{\partial \psi}{\partial y} \right) + M_\psi \\
\frac{\partial \chi}{\partial t} &= -\left\{ \frac{1}{a \cos \varphi} \frac{\partial \chi}{\partial z} + v \nu \right\} - \frac{\partial}{\partial c} \left\{ \frac{\partial \chi}{\partial c} \right\} + \frac{\partial}{\partial z} \left( \frac{\partial \chi}{\partial z} \right) + M_\chi
\end{align*}
\]

Physical model

Mathematical description

Discretization / algorithm

Domain science (incl. applied mathematics)

\[
\text{lap}(i,j,k) = -4.0 \times \text{data}(i,j,k) + \\
\text{data}(i+1,j,k) + \text{data}(i-1,j,k) + \\
\text{data}(i,j+1,k) + \text{data}(i,j-1,k) + \text{data}(i+1,j,k) + \text{data}(i-1,j,k) + \text{data}(i,j+1,k) + \text{data}(i,j-1,k) + \text{data}(i+1,j,k) + \text{data}(i-1,j,k) + \text{data}(i,j+1,k) + \text{data}(i,j-1,k);
\]

Code / implementation

“Port” serial code to supercomputers

> vectorize 
> parallelize 
> petascaling 
> exascaling 
> ...

Code compilation

Computer engineering
(& computer science)
\[
\frac{\partial u}{\partial t} = -\left\{ \frac{1}{a \cos \varphi} \frac{\partial E}{\partial x} - v v_x \right\} - \frac{1}{\rho \cos \varphi} \left( \frac{\partial p}{\partial x} - \frac{1}{\sqrt{T}} \frac{\partial q}{\partial x} \right) + M_u
\]

\[
\frac{\partial v}{\partial t} = -\left\{ \frac{1}{a \cos \varphi} \frac{\partial E}{\partial y} + u v_y \right\} - \frac{1}{\rho \cos \varphi} \left( \frac{\partial p}{\partial y} - \frac{1}{\sqrt{T}} \frac{\partial q}{\partial y} \right) + M_v
\]

\[
\frac{\partial w}{\partial t} = -\left\{ \frac{1}{a \cos \varphi} \frac{\partial E}{\partial z} + v v_z \right\} - \frac{1}{\rho \cos \varphi} \left( \frac{\partial p}{\partial z} + \frac{g \rho}{\sqrt{T}} \frac{\partial q}{\partial z} \right) + M_w
\]

\[
\frac{\partial q}{\partial t} = -\left\{ \frac{1}{a \cos \varphi} \frac{\partial E}{\partial x} + v v_y \right\} - \frac{1}{\rho \cos \varphi} \left( \frac{\partial p}{\partial y} - \frac{1}{\sqrt{T}} \frac{\partial q}{\partial y} \right) + M_q
\]

\[
\frac{\partial q}{\partial t} = -\left\{ \frac{1}{a \cos \varphi} \frac{\partial E}{\partial y} + v v_x \right\} - \frac{1}{\rho \cos \varphi} \left( \frac{\partial p}{\partial x} - \frac{1}{\sqrt{T}} \frac{\partial q}{\partial x} \right) + M_q
\]

\[
\frac{\partial q}{\partial t} = -\left\{ \frac{1}{a \cos \varphi} \frac{\partial E}{\partial z} + v v_z \right\} - \frac{1}{\rho \cos \varphi} \left( \frac{\partial p}{\partial z} + \frac{g \rho}{\sqrt{T}} \frac{\partial q}{\partial z} \right) + M_q
\]

\[
\frac{\partial t}{\partial t} = \frac{1}{2} \left( \frac{\partial t}{\partial k} \right) + \frac{1}{2} \left( \frac{\partial t}{\partial l} \right) + \frac{1}{2} \left( \frac{\partial t}{\partial m} \right) + M_t
\]

Mathematical description

Discretization / algorithm

Domain science (incl. applied mathematics)

Tools & Libraries

Code / implementation

Code compilation

Architectural options / design

Optimal algorithm

Auto tuning

Computer engineering (& computer science)
\[
\begin{aligned}
\frac{\partial u}{\partial t} &= - \frac{1}{\rho \cos \varphi} \frac{\partial E_h}{\partial x} - v \frac{\partial u}{\partial x} - \frac{1}{\rho \cos \varphi} \left( \frac{\partial \rho u}{\partial x} - \frac{1}{\sqrt{\gamma} \sin \varphi} \frac{\partial p u}{\partial x} \right) + \mathcal{M}_u \\
\frac{\partial v}{\partial t} &= - \frac{1}{\rho \sin \varphi} \left( \frac{\partial E_h}{\partial y} + v \frac{\partial u}{\partial y} \right) - \frac{1}{\rho \sin \varphi} \left( \frac{\partial \rho u}{\partial y} + \frac{\rho \cos \varphi}{\sqrt{\gamma} \sin \varphi} \frac{\partial p u}{\partial y} \right) + \mathcal{M}_v \\
\frac{\partial p}{\partial t} &= - \frac{1}{\rho \cos \varphi} \left( \frac{\partial E_h}{\partial z} + v \frac{\partial u}{\partial z} \right) - \frac{1}{\rho \sin \varphi} \left( \frac{\partial \rho u}{\partial z} + \frac{\rho \cos \varphi}{\sqrt{\gamma} \sin \varphi} \frac{\partial p u}{\partial z} \right) + \mathcal{M}_p \\
\frac{\partial T}{\partial t} &= - \frac{1}{\rho c_p} \left( \frac{\partial E_h}{\partial z} + v \frac{\partial u}{\partial t} \right) - \frac{1}{c_p} \left( \frac{\partial \rho u}{\partial z} + \frac{\rho \cos \varphi}{\sqrt{\gamma} \sin \varphi} \frac{\partial p u}{\partial z} \right) + \mathcal{M}_T \\
\frac{\partial S}{\partial t} &= - \frac{1}{\rho \cos \varphi} \left( \frac{\partial E_h}{\partial y} + v \frac{\partial u}{\partial y} \right) - \frac{1}{\rho \sin \varphi} \left( \frac{\partial \rho u}{\partial y} + \frac{\rho \cos \varphi}{\sqrt{\gamma} \sin \varphi} \frac{\partial p u}{\partial y} \right) + \mathcal{M}_S \\
\end{aligned}
\]

Mathematical description

Discretization / algorithm

Code / implementation

Tools & Libraries

Computer engineering (& computer science)

Domain science

Architectural options / design

Code compilation

Optimal algorithm

Auto tuning

PASC co-design projects

Model development based on Python or equivalent dynamic language
Thank you!
Thank you to a great team!

- Oliver Fuhrer (Meteo Swiss)
- Tobias Gysi and David Müller (Supercomputing Systems)
- Xavier Lapillone and Carlos Osuna (C2SM @ ETH Zurich)
- Will Saywer, Mauro Bianco, Ugo Vareto, Ben Cummings, Gille Fourestey (CSCS)
- Tim Schröder and Peter Messmer (NVIDIA)
- Ulli Schätler and Michael Baldauf (Deutscher Wetterdienst)
- Many helpful discussions with Christoph Schär and Torsten Höfler