Modeling Systems at the end of Dennard Scaling

PASC18 Session MS-42
Basel Switzerland

V. Balaji

NOAA/GFDL and Princeton University

4 July 2018
Outline

1. Hardware evolution at the end of Dennard scaling
   - The end of Dennard scaling
   - Specialized and commodity computing
   - Increased concurrency, slower arithmetic
   - Deep learning is an industry driver

2. Approaches to modeling post-Dennard
   - We’ll always have “physics”
   - Generate low-dimensional representations from higher-dimensional
   - Models as task graphs
   - Energy and time to solution

3. Coarse-grain concurrency
   - Concurrent nesting
   - Ice-ocean boundary
   - Chemistry, dust, ...

4. Summary
Outline

1. Hardware evolution at the end of Dennard scaling
   - The end of Dennard scaling
   - Specialized and commodity computing
   - Increased concurrency, slower arithmetic
   - Deep learning is an industry driver

2. Approaches to modeling post-Dennard
   - We’ll always have “physics”
   - Generate low-dimensional representations from higher-dimensional
   - Models as task graphs
   - Energy and time to solution

3. Coarse-grain concurrency
   - Concurrent nesting
   - Ice-ocean boundary
   - Chemistry, dust, ...

4. Summary
Moore’s Law and End of Dennard scaling

Power and Heat Problems Led to Multiple Cores and Prevent Further Improvements in Speed

Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore
Source: Chuck Moore, Data Processing in Exascale-Class Systems; April 27, 2011; Salt Lake Conference on High-Speed Computing

Figure courtesy Moore 2011: *Data processing in exascale-class systems.*

- Processor concurrency: Intel Xeon-Phi.
- Fine-grained thread concurrency: Nvidia GPU.
- HPCG/HPL ratio is a measure of “percent of peak” (Dongarra and Heroux 2013).
- All recent HPC acquisitions in climate/weather have been on conventional Intel Xeon (see Balaji et al 2017).

<table>
<thead>
<tr>
<th>Site</th>
<th>Computer</th>
<th>Cores</th>
<th>HPL Rmax (Pflops)</th>
<th>HPL Rank</th>
<th>HPCG (Pflops)</th>
<th>HPCG/HPL</th>
<th>% of Peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSCC / Guangzhou</td>
<td>Tianhe-2 NUDT, Xeon 12C 2.2GHz + Intel Xeon Phi 57C + Custom</td>
<td>3,120,000</td>
<td>33.9</td>
<td>1</td>
<td>.632</td>
<td>1.8%</td>
<td>1.1%</td>
</tr>
<tr>
<td>RIKEN Advanced Inst for Comp Sci</td>
<td>K computer Fujitsu SPARC64 VIII fx BC + Custom</td>
<td>705,024</td>
<td>10.5</td>
<td>4</td>
<td>.461</td>
<td>4.4%</td>
<td>4.1%</td>
</tr>
<tr>
<td>DOE/OS Oak Ridge Nat Lab</td>
<td>Titan, Cray XK7 AMD 16C + Nvidia Kepler GPU 14C + Custom</td>
<td>560,640</td>
<td>17.6</td>
<td>2</td>
<td>.322</td>
<td>1.8%</td>
<td>1.2%</td>
</tr>
<tr>
<td>DOE/OS Argonne Nat Lab</td>
<td>Mira BlueGene/Q, Power BQC 16C 1.6GHz + Custom</td>
<td>786,432</td>
<td>8.59</td>
<td>5</td>
<td>.167</td>
<td>1.9%</td>
<td>1.7%</td>
</tr>
<tr>
<td>Swiss CSCS</td>
<td>Piz Daint, Cray XC30, Xeon 8C + Nvidia Kepler 14C + Custom</td>
<td>115,984</td>
<td>6.27</td>
<td>6</td>
<td>.105</td>
<td>1.7%</td>
<td>1.3%</td>
</tr>
<tr>
<td>Leibniz Rechenzentrum</td>
<td>SuperMUC, Intel 8C + IB</td>
<td>147,456</td>
<td>2.90</td>
<td>14</td>
<td>.0833</td>
<td>2.9%</td>
<td>2.6%</td>
</tr>
<tr>
<td>DOE/OS LBNL</td>
<td>Edison, Cray XC30, Xeon, 12C, 2.4GHz + Custom</td>
<td>133,824</td>
<td>1.65</td>
<td>24</td>
<td>.0786</td>
<td>4.8%</td>
<td>3.1%</td>
</tr>
<tr>
<td>GSIC Center TITech</td>
<td>Tsubame 2.5 Xeon 6C, 2.93GHz + Nvidia K20x + IB</td>
<td>76,032</td>
<td>2.78</td>
<td>15</td>
<td>.073</td>
<td>2.6%</td>
<td>1.3%</td>
</tr>
<tr>
<td>Max-Planck</td>
<td>iDataPlex Xeon 10C, 2.8GHz + IB</td>
<td>65,320</td>
<td>1.28</td>
<td>34</td>
<td>.061</td>
<td>4.8%</td>
<td>4.2%</td>
</tr>
<tr>
<td>CEA/TGCC-GENCI</td>
<td>Curie time nodes Bullix BS10 Intel Xeon 8C 2.7 GHz + IB</td>
<td>77,184</td>
<td>1.36</td>
<td>33</td>
<td>.051</td>
<td>3.8%</td>
<td>3.1%</td>
</tr>
<tr>
<td>Exploration and Production Eni S.p.A.</td>
<td>HPC2, Intel Xeon 10C 2.8 GHz + Nvidia Kepler 14C + IB</td>
<td>62,640</td>
<td>3.00</td>
<td>12</td>
<td>.0489</td>
<td>1.6%</td>
<td>1.2%</td>
</tr>
</tbody>
</table>
The inexorable triumph of commodity computing

The "Navier-Stokes Computer" of 1986

“The Navier-Stokes computer (NSC) has been developed for solving problems in fluid mechanics involving complex flow simulations that require more speed and capacity than provided by current and proposed Class VI supercomputers. The machine is a parallel processing supercomputer with several new architectural elements which can be programmed to address a wide range of problems meeting the following criteria: (1) the problem is numerically intensive, and (2) the code makes use of long vectors.”

Nosenchuck and Littman (1986)
The Caltech "Cosmic Cube" (1986)

Figure 2. The 128-node Mark II system at Caltech (a) showing, in the back view (b), the cabling that implements the hypercube interconnect.

“Caltech is at its best blazing new trails; we are not the best place for programmatic research that dots i’s and crosses t’s”. Geoffrey Fox, pioneer of the Caltech Concurrent Computation Program, in 1986.
Beowulf clusters

Introduction

While our first Beowulf-style parallel computer isn't built out of the most impressive hardware, we got tired of fighting for funding and went ahead with what we could find. Much like the classic *Tale of Stone Soup*, many individuals contributed to the existing machine*. Because of a complete lack of funding, we used surplus personal computers donated by individuals from ORNL, the Procurement Dept., Y-12, and K-25, to build a parallel computer system which uses public domain compilers and message passing libraries. This system was built at literally no cost.

![Image](https://www.extremelinux.info/stonesoup/)

We are adding more nodes every week. Click [here](https://www.extremelinux.info/stonesoup/) to donate your personal computer equipment to the Stone SouperComputer. And be sure to tell your friends.

People are often interested in the price-to-performance ratio of their computer systems. Since our cost was approximately nothing, any performance results in a zero price-to-performance ratio:

\[
\frac{\text{Price}}{\text{Performance}} = \frac{-0}{\text{anything}} \to 0
\]

Performance-to-price is more interesting. If we get any performance at all, the performance-to-price ratio goes quickly to infinity.

\[
\frac{\text{Performance}}{\text{Price}} = \frac{\text{anything}}{-0} \to \infty
\]

As soon as you login, we all win!!
Power-8 with NVLink

Figure courtesy IBM.
KNL Overview

Figure courtesy Intel.
Deep learning is a layered NN approach with hidden layers. Figure courtesy NVidia.
Google TPU (Tensor Processing Unit)

Figure courtesy Google.
Google TPU (Tensor Processing Unit)

Hardware pipelining of steps in matrix-multiply. Figure courtesy Google.
Outline

1. Hardware evolution at the end of Dennard scaling
   - The end of Dennard scaling
   - Specialized and commodity computing
   - Increased concurrency, slower arithmetic
   - Deep learning is an industry driver

2. Approaches to modeling post-Dennard
   - We’ll always have “physics”
   - Generate low-dimensional representations from higher-dimensional
   - Models as task graphs
   - Energy and time to solution

3. Coarse-grain concurrency
   - Concurrent nesting
   - Ice-ocean boundary
   - Chemistry, dust, ...

4. Summary
No separation of "large" and "small" scales

Nastrom and Gage (1985).
Generating parameterizations from CRMs and super-parameterization

- Global-scale CRMs (e.g. 7 km simulation on the left) and even super-parameterization using embedded cloud models (right) remain prohibitively expensive.
- Use emulators (genetic programming or DL using GCM-resolution predictors) to emulate columns of a cloud field.

(Courtesy: S-J Lin, NOAA/GFDL).

(Courtesy: D. Randall, CSU; CMMAP).
Examples of DAG parallelism

DAG example: Cholesky Inversion

DAG = Directed Acyclic Graph

Can IFS use this technology?

Source: Stan Tomov, ICL, University of Tennessee, Knoxville

Figure courtesy George Mozdzynski, ECMWF.
prioQueue.enqueue(source, 0)
while prioQueue not empty:
    (node, dist) = prioQueue.dequeueMin()
    if node.distance not set:
        node.distance = dist
        for nbr in node.neighbors:
            d = dist + edgeWeight(node, nbr)
            prioQueue.enqueue(nbr, d)
    else: // node already visited, skip

---

(a) SWARM for DAGs

(b)  

Parent-child relations  Data dependences

(c)  

Order = Distance from source node

---

JPSY comparison across ESMs

<table>
<thead>
<tr>
<th>Model</th>
<th>Machine</th>
<th>Resol</th>
<th>SYPD</th>
<th>CHSY</th>
<th>JPSY</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM4</td>
<td>gaea/c2</td>
<td>$1.2 \times 10^8$</td>
<td>4.5</td>
<td>16000</td>
<td>$8.92 \times 10^8$</td>
</tr>
<tr>
<td>CM4</td>
<td>gaea/c3</td>
<td>$1.2 \times 10^8$</td>
<td>10</td>
<td>7000</td>
<td>$3.40 \times 10^8$</td>
</tr>
</tbody>
</table>

- Comparative measures of capability (SYPD), capacity (CHSY), and energy cost (JPSY) per “unit of science”.
- Can you have codes that are “slower but greener”? Algorithms that are “less accurate but more eco-friendly”?
Outline

1. Hardware evolution at the end of Dennard scaling
   - The end of Dennard scaling
   - Specialized and commodity computing
   - Increased concurrency, slower arithmetic
   - Deep learning is an industry driver

2. Approaches to modeling post-Dennard
   - We’ll always have “physics”
   - Generate low-dimensional representations from higher-dimensional
   - Models as task graphs
   - Energy and time to solution

3. Coarse-grain concurrency
   - Concurrent nesting
   - Ice-ocean boundary
   - Chemistry, dust, ...

4. Summary
The atmospheric radiation component computes radiative transfer of incoming shortwave solar fluxes and outgoing longwave radiation as a function of all radiatively active species in the atmosphere (greenhouse gases, aerosols, particulates, clouds, ...).

- The physics of radiative transfer is relatively well-known, but a full Mie-scattering solution is computationally out of reach.
- Approximate methods (sampling the “line-by-line” calculation into “bands”) have been in use for decades, and “standard” packages like RRTM are available.
- They are still very expensive: typically $\Delta t_{rad} > \Delta t_{phy}$ (in the GFDL models typically 9X). The model is sensitive to this ratio.
- Other methods: stochastic sampling of bands (Pincus and Stevens 2013), neural nets (Krasnopolsky et al 2005)

Challenge: can we exploit “cheap flops” to set $\Delta t_{rad} = \Delta t_{phy}$?
Traditional coupling sequence

Radiation timestep much longer than physics timestep.
(Figure courtesy Rusty Benson, NOAA/GFDL.)
Radiation executes on physics timestep from lagged state.
(Figure courtesy Rusty Benson, NOAA/GFDL).
Concurrent coupling sequence using pelists

Requires MPI communication between physics and radiation. (Figure courtesy Rusty Benson, NOAA/GFDL).
Physics and radiation share memory.
(Figure courtesy Rusty Benson, NOAA/GFDL).
Results from climate run

20 year AMIP/SST climate runs have completed on Gaea (Cray XE6).

- Control: 9.25 sYPD
  - $\Delta t_{rad} = 9\Delta t_{phy}$
  - 864 MPI-ranks / 2 OpenMP threads
- Serial Radiation: 5.28 sYPD
  - $\Delta t_{rad} = \Delta t_{phy}$
  - 864 MPI-ranks / 2 OpenMP threads
- Concurrent Radiation: 5.90 sYPD
  - $\Delta t_{rad} = \Delta t_{phy}$
  - 432 MPI-ranks / 4 OpenMP threads (2 atmos + 2 radiation)
  - Can get back to 9 sYPD at about $\sim 2700$ cores (roughly 1.6X).

Comparison of Concurrent Radiation to Control

- Climate is similar
- TOA balance is off by $\sim 4\ W/\ m^2$, mostly in the short wave, but easily retuned when ready to deploy

See Balaji et al (2016).
Lee vortices off Hawaii under two-way nesting

- 72 hr forecast from 1 Aug 2010 00Z with real topography
- Showing Vorticity $\times 10^5$

Figure courtesy Lucas Harris and S-J Lin, NOAA/GFDL.
Concurrent two-way nesting

Typical nesting protocols force serialization between fine and coarse grid timestepping, since the $C^*$ are estimated by interpolating between $C^n$ and $C^{n+1}$.

We enable concurrency by instead estimating the $C^*$ by extrapolation from $C^{n-1}$ and $C^n$, with an overhead of less than 10%. (See Harris and Lin 2012 for details.)
Concurrent coupling: possible stability issues

Evidence of Lagged Stress-Inertial Coupling Instability in Sea-Ice Thickness

January 1, Year 20

$\Delta t_{cpl} = 3600 \text{ s}$

$\Delta t_{cpl} = 7200 \text{ s}$

Sequentially coupled data-driven ice-ocean model

Hallberg (2014, *Clivar Exchanges*)

Figure courtesy Bob Hallberg (GFDL).
Concurrent coupling: possible stability issues

Explosive Sea-Ice Growth as a Manifestation of a Sea Ice-Ocean Coupling Instability with KPP

Figure courtesy Bob Hallberg (GFDL).

V. Balaji (balaji@princeton.edu)
Sequential coupling

\[ Q(SST,T_i,T_a), \tau(u_o,u_i,u_a) \]

SST, \( u_o \)

Ocean Dynamics | Ocean Thermo
---|---
| ---

\[ t^n \quad \Delta t_{\text{coupled}} \quad t^{n+1} \]

Figure courtesy Alistair Adcroft, Princeton and GFDL.
Ice-ocean boundary

Concurrent coupling

Figure courtesy Alistair Adcroft, Princeton and GFDL.
Embedding SIS2

Figure courtesy Alistair Adcroft, Princeton and GFDL.
Staggered-concurrent coupling

\[ \tau(u_o, u_i, u_a) \]

\[ Q(SST, T_i, T_a) \]

\[ u_o \]

\[ SST \]

\[ \Delta t_{\text{coupled}} \]

\[ t^n \]

\[ t^{n+1} \]

Figure courtesy Alistair Adcroft, Princeton and GFDL.
Sequential coupling + Adams-Bashforth

Figure courtesy Alistair Adcroft, Princeton and GFDL.
Ice-ocean boundary

Concurrent coupling + AB

Figure courtesy Alistair Adcroft, Princeton and GFDL.
Ice-ocean boundary

Staggered-concurrent coupling + AB

\[ \text{Atmos. Thermo} \]
\[ \text{Ice Thermo} \]
\[ \text{Atmos. Thermo} \]
\[ \text{Atmosphere Dynamics} \]
\[ \text{Ice Dynamics} \]
\[ \text{Atmos. Thermo} \]
\[ \text{Atmosphere Dynamics} \]
\[ \text{Ice Dynamics} \]
\[ \text{Ice Thermo} \]
\[ \text{Ice Dynamics} \]

\[ \tau(u_o, u_i, u_a) \]

\[ Q(SST, T_i, T_a) \]

\[ u_o \]

\[ \text{SST} \]

\[ \text{SST}^* \]

\[ \text{SST} \]

\[ \text{SST} \]

\[ \text{SST} \]

\[ t^{n-1} \]

\[ t^n \]

\[ \Delta t_{coupled} \]

\[ t^{n+1} \]

Figure courtesy Alistair Adcroft, Princeton and GFDL.
Managing Chemistry Dependencies/Feedback (1)

Incoming:
- Dynamics: Lat-lon wind speeds
- Physics: Planetary boundary layer depth
- Radiation: Extinction values

Outgoing:
- Chemistry: Tracer tendencies.
- Physics: Added to the vertical diffusion tendencies and exchanged with the land.
  Input to the moist physics calculation.

Chemistry:
- Compute tracer surface fluxes (dust, sea salt, ...).
- Compute carbon aerosols and sulfur chemistry.
- Compute stratospheric chemistry.

Figure courtesy Ray Menzel, Engility and GFDL.
Outline

1. Hardware evolution at the end of Dennard scaling
   - The end of Dennard scaling
   - Specialized and commodity computing
   - Increased concurrency, slower arithmetic
   - Deep learning is an industry driver

2. Approaches to modeling post-Dennard
   - We’ll always have “physics”
   - Generate low-dimensional representations from higher-dimensional
   - Models as task graphs
   - Energy and time to solution

3. Coarse-grain concurrency
   - Concurrent nesting
   - Ice-ocean boundary
   - Chemistry, dust, ...

4. Summary

V. Balaji (balaji@princeton.edu)
No scale separation implies a catastrophic cascade of dimensionality: we’re off by $10^{10}$ from required flops, Schneider et al (2017).

Moore’s law has taken us from the von Neumann model to the “sea of functional units” (Kathy Yelick). Not easy to understand, predict or program performance.

... but the “free lunch” decades are over, they’ve come to take away your plates.

Coarse-grained parallelism is an area in the current effort to reclaim performance from the encroaching “sea”.

The “component” abstraction still may let us extract some benefits out of the machines of this era:

- sharing of the wide thread space.
- distribute components among heterogeneous hardware?
- concerns about stability, conservation, and accuracy.

