Introduction to Parallel Processing

ACF Spring HPC Training Workshop
Match 15-16, 2016
Kwai Wong
Acknowledgements:

- Support from NSF, UTK, JICS, NICS
- Efforts from many colleagues, collaborators, and students
- Credits to many researchers and industrial practitioners for a lot of materials that I use in this talk, plenty of new science, new technologies and applications in HPC

Contents

- Landscape of Supercomputers
- Performance Ranking : Top500, HPL
- Supercomputers => Big Science + Big Data
- Programming Model on High Performance Computers
- General Practices
Joint Institute for Computational Sciences

- JICS is a joint research center between UTK and ORNL since 1991 to advance computational sciences activities
- Joint Faculty, research staff, National Institutes for Computational Sciences
- Projects: Kraken, RDAV, Keeneland, Beacon, XSEDE, ACF
- Total JICS funding > $100M
NICS – beacon (Xeon Phi), darter (XC30, kraken-E)

**WORLD RECORD!**

“Beacon” at NICS

Intel® Xeon® + Intel Xeon Phi™ Cluster

First to Deliver

2.499 GigaFLOPS / Watt

71.4% efficiency

#1 on current Green500

**kraken**: 1st Academic PetaFLOPS Computer (3rd 2009), 100 Cabinets, 112896 cores
ORNL is the U.S. Department of Energy’s largest science and energy laboratory

Oak Ridge Leadership Computing Facility (OLCF)

- World’s premier computing facility
- Nation’s largest concentration of open source materials research
- Nation’s most diverse energy portfolio
- $1B+ Spallation Neutron Source project
- Managing the $1B+ U.S. ITER project
ORNL’s “Titan” Hybrid System: Cray XK7 with AMD Opteron and NVIDIA Tesla processors

SYSTEM SPECIFICATIONS:
- Peak performance of 27 PF
  - 24.5 Pflop/s GPU + 2.6 Pflop/s AMD
- 18,688 Compute Nodes each with:
  - 16-Core AMD Opteron CPU
  - 14-Core NVIDIA Tesla “K20x” GPU
  - 32 GB + 6 GB memory
- 512 Service and I/O nodes
- 200 Cabinets
- 710 TB total system memory
- Cray Gemini 3D Torus Interconnect
- 9 MW peak power

✓ (CPU) 2.26 x 4 x 18688 = 2.392 ; (GPU) 1.31 x 18688 x 14 = 24.27 PF, Peak=26.67
✓ 17.5 PFLOPS (HPL) 64.8% ; ~ 10 times faster than jaguar; 9 Megawatt,
✓ 900 W/apartment – 10000 apartments !! --- Currently No. 5 in the world
Sumway: Fastest Computer: TOP500

- Wuxi
- June 2016
- 15.3 MW
- 93 PF
Sunway - Wuxi - China
Top500 – Nov. 2017 – top500 list every 6 months
Solving a $Ax=b$ : $A$ is dense $NxN$ Matrix ; MM

<table>
<thead>
<tr>
<th>#</th>
<th>Site</th>
<th>Manufacturer</th>
<th>Computer</th>
<th>Country</th>
<th>Cores</th>
<th>$R_{max}$ [Petaflop]</th>
<th>Power [MW]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>National Supercomputing Center in Wuxi</td>
<td>NRCPC</td>
<td>Sunway TaihuLight</td>
<td>China</td>
<td>10,649,600</td>
<td>93.0</td>
<td>15.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NRCPC Sunway SW26010, 260C 1.45GHz</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>National University of Defense Technology</td>
<td>NUDT</td>
<td>Tianhe-2</td>
<td>China</td>
<td>3,120,000</td>
<td>33.9</td>
<td>17.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NUDT TH-IVB-FEP, Xeon 12C 2.2GHz, InteXeon Phi</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Swiss National Supercomputing Centre (CSCS)</td>
<td>Cray</td>
<td>Piz Daint</td>
<td>Switzerland</td>
<td>361,760</td>
<td>19.6</td>
<td>2.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cray XC50, Xeon E5 12C 2.6GHz, Aries, NVIDIA Tesla P100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Japan Agency for Marine-Earth Science and Technology</td>
<td>ExaScaler</td>
<td>Gyoukou</td>
<td>Japan</td>
<td>19,860,000</td>
<td>19.1</td>
<td>1.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ZettaScaler-2.2 HPC System, Xeon 16C 1.3GHz, IB-EDR, PEZ-Y-SC2 700Mhz</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Oak Ridge National Laboratory</td>
<td>Cray</td>
<td>Titan</td>
<td>USA</td>
<td>560,640</td>
<td>17.6</td>
<td>8.21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cray XK7, Opteron 16C 2.2GHz, Gemini, NVIDIA K20x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Lawrence Livermore National Laboratory</td>
<td>IBM</td>
<td>Sequoia</td>
<td>USA</td>
<td>1,572,864</td>
<td>17.2</td>
<td>7.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BlueGene/Q, Power BQC 16C 1.6GHz, Custom</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Los Alamos NL / Sandia NL</td>
<td>Cray</td>
<td>Trinity</td>
<td>USA</td>
<td>979,968</td>
<td>14.1</td>
<td>3.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Lawrence Berkeley National Laboratory</td>
<td>Cray</td>
<td>Cori</td>
<td>USA</td>
<td>622,336</td>
<td>14.0</td>
<td>3.94</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cray XC40, Intel Xeons 7250 68C 1.4 GHz, Aries</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>JCAHPC Joint Center for Advanced HPC</td>
<td>Fujitsu</td>
<td>Oakforest-PACS</td>
<td>Japan</td>
<td>556,104</td>
<td>13.6</td>
<td>2.72</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PRIMERGY CX1640 M1, Intel Xeons Phi 7250 68C 1.4 GHz, OmniPath</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>RIKEN Advanced Institute for Computational Science</td>
<td>Fujitsu</td>
<td>K Computer</td>
<td>Japan</td>
<td>795,024</td>
<td>10.5</td>
<td>12.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SPARC64 VIIIfx 2.0GHz, Tofu Interconnect</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Jaguar: 2009 World’s Most Powerful Computer
www.olcf.ornl.gov

<table>
<thead>
<tr>
<th></th>
<th>jaguar XT4</th>
<th>jaguarpf XT5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak Performance</td>
<td>263.16 TFLOPS</td>
<td>2.33 PFLOPS</td>
</tr>
<tr>
<td>System Memory</td>
<td>61 TB</td>
<td>292 TB</td>
</tr>
<tr>
<td>Disk Space</td>
<td>750 TB</td>
<td>10,000 TB</td>
</tr>
<tr>
<td>Disk Bandwidth</td>
<td>44 GB/s</td>
<td>240 GB/s</td>
</tr>
<tr>
<td>Interconnect Bandwidth</td>
<td>157 TB/s</td>
<td>374 TB/s</td>
</tr>
</tbody>
</table>
## TOP500 List - November 2009 (1-100)

*R_{\text{max}}* and *R_{\text{peak}}* values are in TFlops. For more details about other fields, check the TOP500 description.

Power data in KW for entire system

<table>
<thead>
<tr>
<th>Rank</th>
<th>Site</th>
<th>Computer/Year Vendor</th>
<th>Cores</th>
<th>R_{\text{max}}</th>
<th>R_{\text{peak}}</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Oak Ridge National Laboratory, United States</td>
<td>Jaguar - Cray XT5-HE Opteron Six Core 2.6 GHz / 2009 Cray Inc.</td>
<td>224162</td>
<td>1759.00</td>
<td>2331.00</td>
<td>6950.60</td>
</tr>
<tr>
<td>2</td>
<td>DOE/NNSA/LANL, United States</td>
<td>Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband / 2009 IBM</td>
<td>122400</td>
<td>1042.00</td>
<td>1375.78</td>
<td>2345.50</td>
</tr>
<tr>
<td>3</td>
<td>National Institute for Computational Sciences/University of Tennessee, United States</td>
<td>Kraken XT5 - Cray XT5-HE Opteron Six Core 2.6 GHz / 2009 Cray Inc.</td>
<td>98928</td>
<td>831.70</td>
<td>1028.85</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Forschungszentrum Juelich (FZJ), Germany</td>
<td>JUGENE - Blue Gene/P Solution / 2009 IBM</td>
<td>294912</td>
<td>825.50</td>
<td>1002.70</td>
<td>2268.00</td>
</tr>
<tr>
<td>5</td>
<td>National SuperComputer Center in Tianjin/NUDT, China</td>
<td>Tianhe-1 - NUDT TH-1 Cluster, Xeon E5540/E5450, ATI Radeon HD 4870 2, Infiniband / 2009 NUDT</td>
<td>71680</td>
<td>563.10</td>
<td>1206.19</td>
<td></td>
</tr>
</tbody>
</table>
Performance Development

Hear more about this and the latest data at our BoF following at 5:15pm.
Numbers: Lots of Them: bit, byte, FLOP (S)

- Core: computing unit: processor
- Dual core machine (Intel or AMD CPU): a CPU with 2 cores, each core is a 2.4 GHz computing unit with 2GB of RAM (memory in the processor not disk space)
- Binary bits (b): “0” or “1” , 1 Byte (B) = 8 bits
- Binary number: 11111111 = \(2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0\) = \(2^8 - 1\) = 255 !!
- 32 bits machine or operating system => largest integer (all positive) = \(2^{32} - 1\) = (4,294,967,296 -1) or range of integer = \(-(2^{31})\) to \(2^{31} -1\)
- 64 bits machine or operating system => range of integer = \(-(2^{63})\) to \(2^{63} -1\)
- Kilo (K) = \(10^3\) (or \(2^{10}\)); Mega (M) = \(10^6\) (or \(2^{20}\)); Giga (G) = \(10^9\) (or \(2^{30}\)); Tera (T billion) = \(10^{12}\) (or \(2^{40}\)); Peta (P) = \(10^{15}\) (or \(2^{50}\))
- Floating Point Operation (+, -, /, *): \((10.1 + 0.1) \times 1.0 / 2.0 = 5.1\) => 3 FLOP
- FLOPS = FLOP per second :: 1 PetaFLOPS (kraken) = \(10^{15}\) FLOP in one second
- FLOPS in a core = (clock rate) \times (floating point operation in one clock cycle)
- Peak Rate = (FLOPS in one compute unit, core) \times (no. of core)
HPL (High Performance Linpack): Solving $Ax = b$

\[
2x_1 + 2x_2 + 2x_3 = 1 \\
3x_1 + 4x_2 + 5x_3 = 2 \\
4x_1 + 6x_2 + 7x_3 = 3.
\]

\[
A = \begin{bmatrix}
2 & 2 & 2 \\
3 & 4 & 5 \\
4 & 6 & 7
\end{bmatrix}, \quad b = \begin{bmatrix}
1 \\
2 \\
3
\end{bmatrix}
\]

Total operation count for Gaussian elimination with backward substitution

\[
\frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{7}{6}n.
\]
Jaguar (ORNL): World Fastest Computer, 1.759 PF (2009)

- **FLOPS** – Floating Point Operation Per Second
- **GFLOPS** = $10^9$ FLOPS ; **TFLOPS** = $10^{12}$ ; **PFLOPS** = $10^{15}$
- **FLOPS** = (clock rate) $\times$ (floating point operation in one clock cycle)
- **Peak Rate** = (FLOPS in one CPU) $\times$ (no. of CPU)
- Cray XT5 one core AMD Opteron:
  - $R_{\text{peak}} : (2.6 \text{ GHz}) \times (4) \times (224162 \text{ cores}) = 2331284 \text{ GFLOPS}$
  - $R_{\text{max}} : 1759000 \text{ GFLOPS} \Rightarrow 75.4\%$ of peak

**jaguar: What does it do?**

- **Solve a very big system of equations**: $Ax = b$ using a standard benchmark C program (HPL)
- **$N_{\text{max}}$**: Size of $A$ for HPL (Solve $Ax=b$) = 5474272
- **Total Memory needed** = ($N_{\text{max}}$) x ($N_{\text{max}}$) x (8 Bytes) = 239741 GB
- **Memory needed per core** = 1.07 GB
- **Elapse Time** : $2(N_{\text{max}})(N_{\text{max}})(N_{\text{max}})/3/R_{\text{max}} \approx 13$ hrs
Computer Benchmark (HPL) - Big Science, Big Memory Storage

- **HPL** - Solve a system of equations : $Ax = b$, a standard benchmark C program to rank the top500 computers

- Size of matrix $A =$ Memory used on a computer
- $A = (N_{\text{max}}) \times (N_{\text{max}}) \times (8 \text{ Bytes}) = 239741 \text{ GB (on jaguar)}$

- **Jaguar** : $N_{\text{max}}=5474272$, Memory = 240 TB, $\sim 1.07 \text{ GB/core}$

- **Elapse Time** : $2(N_{\text{max}})(N_{\text{max}})(N_{\text{max}})/3/R_{\text{max}} \sim \sim 13 \text{ hrs (jaguar)}$

- **Titan** $\sim 10$ times faster : $N_{\text{max}} \sim 8000000 : 1.7 \text{ GB / core; titan} \sim 20 \text{ hrs, 65\% of peak performance}$
<table>
<thead>
<tr>
<th></th>
<th>ORNL Titan</th>
<th>NUDT Tianhe-2</th>
<th>Sunway TaihuLight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theoretical Peak</td>
<td>27 Pflop/s =</td>
<td>54.9 Pflop/s =</td>
<td>125.4 Piflop/s = CPEs + MPEs</td>
</tr>
<tr>
<td></td>
<td>(2.6 CPU + 24.5 GPU) Piflop/s</td>
<td>(6.75 CPU + 48.14 Coprocessor) Piflop/s</td>
<td>Cores per Node = 256 CPEs + 4 MPEs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Supemode = 256 Nodes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>System = 160 Supernodes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cores = 260 * 256 * 160 = 10.6M</td>
</tr>
<tr>
<td>HPL Benchmark Flop/s</td>
<td>17.6 Piflop/s</td>
<td>30.65 Piflop/s</td>
<td>93 Piflop/s</td>
</tr>
<tr>
<td>HPL % Peak</td>
<td>65.19%</td>
<td>55.83%</td>
<td>74.16%</td>
</tr>
<tr>
<td>HPCG Benchmark</td>
<td>0.322 Piflop/s</td>
<td>0.580 Piflop/s</td>
<td>0.371 Piflop/s</td>
</tr>
<tr>
<td>HPCG % Peak</td>
<td>1.2%</td>
<td>1.1%</td>
<td>0.30%</td>
</tr>
<tr>
<td>Compute Nodes</td>
<td>18,688</td>
<td>16,000</td>
<td>40,960</td>
</tr>
<tr>
<td>Node</td>
<td>AMD Optron Interlagos (16 cores, 2.2 GHz) plus Nvidia Tesla K20x (14 cores, .732 GHz)</td>
<td>2 - Intel Ivy Bridge (12 cores, 2.2 GHz) plus 3 - Intel Xeon Phi (57 cores, 1.1 GHz)</td>
<td>256 CPEs + 4 MPEs</td>
</tr>
<tr>
<td>Sockets</td>
<td>18,688 Interlagos + 18,688 Nvidia boards</td>
<td>32,000 Ivy Bridge + 48,000 Xeon Phi boards</td>
<td>40,960 nodes with 256 CPEs and 4 MPEs per node</td>
</tr>
<tr>
<td>Node peak performance</td>
<td>1.4508 Tiflop/s =</td>
<td>3.431 Tiflop/s =</td>
<td>3.06 Tiflop/s</td>
</tr>
<tr>
<td></td>
<td>(.1408 CPU + 1.31 GPU) Tiflop/s</td>
<td>(2*.2112 CPU + 3*.1003 Coprocessor) Tiflop/s</td>
<td>CPE: 8 flops/core/cycle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(1.45 GHz<em>8</em>256 = 2.969 Tiflop/s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MPE (2 pipelines)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2<em>4</em>8 flops/core/cycle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(1.45 GHz*1= 0.0928 Tiflop/s)</td>
</tr>
<tr>
<td>Node Memory</td>
<td>32 GB CPU + 6 GB GPU</td>
<td>64 GB CPU + 3*8 GB Coprocessor</td>
<td>32 GB per node</td>
</tr>
<tr>
<td>System Memory</td>
<td>.710 PB = (.598 PB CPU and .112 PB GPU)</td>
<td>1.4 PB = (1.024 PB CPU and .384 PB Coprocessor)</td>
<td>1.31 PB (32 GB*40,960 nodes)</td>
</tr>
<tr>
<td>Configuration</td>
<td>4 nodes per blade, 24 blades</td>
<td>2 nodes per blade, 16 blades per</td>
<td>Node peak performance is 3.06 Tiflop/s, or 11.7 Gflop/s per core.</td>
</tr>
<tr>
<td>Rank (HPL)</td>
<td>Site</td>
<td>Computer</td>
<td>Cores</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>1 (2)</td>
<td>NSCC / Guangzhou</td>
<td>Tianhe-2 NUDT, Xeon 12C 2.2GHz + Intel Xeon Phi 57C + Custom</td>
<td>3,120,000</td>
</tr>
<tr>
<td>2 (5)</td>
<td>RIKEN Advanced Institute for Computational Science</td>
<td>K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect</td>
<td>705,024</td>
</tr>
<tr>
<td>3 (1)</td>
<td>National Supercomputing Center in Wuxi</td>
<td>Sunway TaihuLight--SW26010, Sunway</td>
<td>10,649,600</td>
</tr>
<tr>
<td>4 (4)</td>
<td>DOE/NNSA/LLNL</td>
<td>Sequoia - IBM BlueGene/Q</td>
<td>1,572,864</td>
</tr>
<tr>
<td>5 (3)</td>
<td>DOE/SC/Oak Ridge Nat Lab</td>
<td>Titan - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x</td>
<td>560,640</td>
</tr>
<tr>
<td>6 (7)</td>
<td>DOE/NNSA/LANL/SNL</td>
<td>Trinity - Cray XC40, Intel E5-2698v3, Aries custom</td>
<td>301,056</td>
</tr>
<tr>
<td>7 (6)</td>
<td>DOE/SC/Argonne National Laboratory</td>
<td>Mira - BlueGene/Q, Power BQC 16C 1.60GHz, Custom</td>
<td>786,432</td>
</tr>
<tr>
<td>8 (11)</td>
<td>TOTAL</td>
<td>Pangea -- Intel Xeon E5-2670, Infiniband FDR</td>
<td>218592</td>
</tr>
<tr>
<td>9 (15)</td>
<td>NASA / Mountain View</td>
<td>Pleiades - SGI ICE X, Intel E5-2680, E5-2680V2, E5-2680V3, Infiniband FDR</td>
<td>185,344</td>
</tr>
<tr>
<td>10 (9)</td>
<td>HLRS/University of Stuttgart</td>
<td>Hazel Hen - Cray XC40, Intel E5-2680v3, Cray Aries</td>
<td>185,088</td>
</tr>
</tbody>
</table>
Big Computer
Big Science
Model base
Compute intensive
Climate Simulations and Weather (Storms) forecast

SPC1 (4640x2880x50, dx=1 km)
WRF Forecast starting at 00Z Thu 03 Jun 2010

www.caps.ou.edu
Simulating the Big One on Kraken
Southern California Earthquake Center

- Biggest Earthquake Simulation on San Andreas Fault, the Big One
- Simulated in a 32 billion grid point subset of the SCEC Community Velocity Model (CVM) V4 with a minimum shear-wave velocity of 500 m/s up to a maximum frequency of 1 Hz.
- 96,000 processor cores used for production runs on Kraken, 2.6 hrs WCT, 53 sustained TeraFlop/s
Materials Science Modeling
Bohmian Dynamics: graphene hydrogenation using DFTB

Separation of quantum and classical degrees of freedom

Quantum (U is on!)  Classical (U is off)
Modeling of Heart and Lung
Air Flow Simulation B747 - Validation
- DreamWorks has a "render farm" of servers made up of about 20,000 processors (HP BladeSystem c-Class server blades).

- The image rendering jobs are broken up into small pieces, distributed out to the server farm, and are later recompiled to create the final images for a film.

- Required a whopping 80 million compute hours to render, 15 million more hours than DreamWorks' last record holder, "The Rise of the Guardians."

- Between 300 and 400 animators worked on "The Croods" over the past three years.

- After completing a film, about 70TB worth of data (things like background art or plants) is stored for future usage in future productions.
Road Map to Exascale Computing

- 1962 (CDC 1604), 1976 (Cray 1), 1982 (XMP), 1988 (YMP), 1994 (T90)
- 1992 – DOE HPCC - High Performance Computing and Communication 3T Initiative – 1 teraflops, 1 terabytes of memory, 1 terabytes/s bandwidth
- 1993 – launch of top500 list, CM5, Intel Paragon, ~100GFLOPS
- 1995 – ASCI – DOE Accelerated Strategic Computing Initiative, intended to do nuclear stockpile simulation
- 1996 – first Terascale computer, ASCII RED SNL
- 1998 – Boewulf, PC cluster – Commodity Components
- DOE – supercomputers, projects – SciDAC, Human Genome project, HER, Climate, INCITE – terascale to petascale
- NSF Track I, II Teragrid, XSEDE -1st petascale
- DOE Leadership Computing Program – CORAL program, Exascale
- National Strategic Computing Initiative NSCI
Do It Yourself : A Typical PC Cluster (1999)

- One server node with dual CPU & SCSI Drive
- 5 Fat worker node with 1 GB RAM
- 16 Worker nodes with 512 MB RAM
- one 24 Port 100Mb Switch, total cost ~$40000
Simple Parallel Computer

Many commodity units connected by a COS interconnect
Modern Supercomputers

Commodity plus Accelerator Today

**Commodity**
- Intel Xeon
- 8 cores
- 3 GHz
- 8\*4 ops/cycle
- 96 Gflop/s (DP)

**Accelerator (GPU)**
- Nvidia K20X “Kepler”
- 2688 “Cuda cores”
- .732 GHz
- 2688\*2/3 ops/cycle
- 1.31 Tflop/s (DP)

From ICL Dr. Jack Dongarra: icl.cs.utk.edu
Example of typical parallel machine

Shared memory programming between processes on a board and a combination of shared memory and distributed memory programming between nodes and cabinets.
Scale to the Future

Over 100% increase in Flop/s for K2M2 Tests

Ride the technology curve

Predicted 80%+ Increase in Flop/s

Kepler (to P100, to V100)
Intel MIC (Landing)

Kepler will implement Virtual Memory Space → Will allow larger problems On GPU/CPU “shared” space
Ride on the Hardware Technology Curve

TACC – Stampede
10 PFLOPS

EVEREST facility
- 35 million pixel, 27-tile PowerWall
- 27 NVIDIA 8800 GTX GPUs, dedicated Linux cluster
- Interactive, large-scale, collaborative data analysis
- 30 feet by 8 feet

ORNL – TITAN
20 PFLOPS

64 cores, ~TFLOPS

Transformational Science: RT Simulation

Kepler (2012), ~TFLOPS

Volta..
Summit: Next Generation Supercomputer at ORNL (Exascale)

Challenges: Power limitation, Scaling application performance

TITAN VS SUMMIT
Compute System Comparison

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>TITAN</th>
<th>SUMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute Nodes</td>
<td>18,688</td>
<td>~3,400</td>
</tr>
<tr>
<td>Processor</td>
<td>(1) 16-core AMD Opteron per node</td>
<td>(Multiple) IBM POWER 9s per node</td>
</tr>
<tr>
<td>Accelerator</td>
<td>(1) NVIDIA Kepler K20x per node</td>
<td>(Multiple) NVIDIA Volta GPUs per node</td>
</tr>
<tr>
<td>Memory per node</td>
<td>32GB (DDR3)</td>
<td>&gt;512GB (HBM+DDR4)</td>
</tr>
<tr>
<td>CPU-GPU Interconnect</td>
<td>PCI Gen2</td>
<td>NVLINK (5-12x PCIe3)</td>
</tr>
<tr>
<td>System Interconnect</td>
<td>Gemini</td>
<td>Dual Rail EDR-IB (23 GB/s)</td>
</tr>
<tr>
<td>Peak Power Consumption</td>
<td>9 MW</td>
<td>10 MW</td>
</tr>
</tbody>
</table>

EXAFLOPS: $10^{18}$

OLCF5: 5-10x Summit
~20 MW

2010
Jaguar: 2.3 PF
Multi-core CPU
7 MW

2013
Titan: 27 PF
Hybrid GPU/CPU
9 MW

2017
Summit: 5-10x Titan
Hybrid GPU/CPUs
10 MW

2022
CORAL System

Capability vs Capacity Computing

**Challenges:** Power limitation, scaling

- **Capability Computing,** Single extreme scale, problem, shortest Time

- **Capacity Computing,** medium scale problems, data engine, analysis

**Figure 3. Growth of Amazon S3 objects.**

**Challenges:** Data Movement, scaling
Big Science, Big Data, Big Iron

“Computational science has become the third pillar of the scientific enterprise, a peer alongside theory and physical experiment.”
(Computational Science: Ensuring America’s Competitiveness, 2005)

“The first great scientific breakthrough of the new century – the decoding of the human genome announced in February 2001 – was a triumph of large-scale computational science.”
(Computational Science: Ensuring America’s Competitiveness, 2005)

Dealing with the Knowns and Unknowns
Uncertainty Quantification – Data Analytics

“As we know there are known knowns.
There are things we know we know.
We also know there are known unknowns.
That is to say, we know there are some things we do not know.
But there are also unknown unknowns.
The ones we don’t know we don’t know,” D. Rumsfeld

Given enough data, can we find the unknowns and predict the knowns?
Four Tiers – Computational Ecosystem

Advanced Computing Architectures
- Emergent Architectures
- Tactical Computing
- Next Generation Computing Systems
- High Performance Networking and Memory

Computing Sciences
- Programming Environments
- Programming Languages
- Software Integration

High Performance Computing

Predictive Simulation Sciences
- Computational Math & Algorithms
- Scientific Computing
- Verification, Validation & Uncertainty Quantification
- Applied Computer Modeling and Analysis

Data Intensive Sciences
- Sciences of Large Data
- Computational Math for Data Analytics
- Real-time Data Access & Analytics
Big Data Predictive Model

- A collection of large data sets that are asymmetric or too large to be processed by traditional tools. Often the data sets are noisy and heterogeneous but in general could be co-related to some significant events.

**Big Data Characterized by**

- **Volume**
  - How much data
- **Velocity**
  - The speed at which data arrives and the speed with which decisions based on it must be made
- **Variety**
  - Heterogeneity of storage platforms, data types, representation, semantic interpretation, and security classification or other distribution limitations
- **Veracity**
  - How trustworthy is the data, what is its uncertainty, and what is the error associated with it
- **Value**
  - What is the data worth

**Challenges include** storage, classification, mining, sharing, visualization..

**Need capacity, infrastructure, domain knowledge + compute, CS, Math.**
Programming Models & Tools Ecosystem:

- Flat file, Excel, CVS
- Database, SQL,
- Distributed DD, HDFS
- Large graph, matrix, SVD
- Storage, I/O, network
- Sensors, big instruments
- Data Mining, searching, compression, neural network, deep learning, smart detection, predictive models, visualization
- Images (picture, neutron, thermal, x-ray...), spatial temporal data, noise, signal, voice, smell, ....
- Healthcare, social, politics, science, finance, agriculture, entertainment, geographic, transportation ....
- Perhaps layman sense?!
Milestones – Capacity (Big data)

- 1973 – Internet was “officially” named
- 1990s Internet widely used
- 1993 Mosaic (NCSA), web browser.. netscape, IE, Mozzilla, Firefox..
- 1995- Google, Amazon
- 1996 – IBM Deep Blue Chess machine, first Terascale, ASCII RED
- 1999 – Grid Computing
- 2000 – Baidu
- 2004 – Facebook, MapReduce
- 2005 – Hadoop
- 2006 deep learning, Geoffrey Hinton, Neural Computing
- Clouds, machine learning framework, GPU
- 2015 – NSCI
- 2015 – NSF - Big Data Hub
Big Data – Transportation
Ph.D Students Needed- (Dr. Han, UTK)
Big Data – Modeling
Auto Pilot, GPS
Spatial Database

Parallelize Range Queries

Evacuation Route Planning

- Only in old plan
- Only in new plan
- In both plans

Shortest Paths

Storing graphs in disk blocks

Source - “From GPS and Virtual Globes to Spatial Computing,” Shashi Shekhar. IEEE Big Data Conference 2015
Big Data Applications: Healthcare

Visually, when scanning through the entire tumor volume, what proportion of the tumor is estimated to represent necrosis. Necrosis is defined as a region within the tumor that does not enhance or shows markedly diminished enhancement, is high on T2W and proton density images, is low on T1W images, and has an irregular border. (Assuming that the entire abnormality may be comprised of: (1) an enhancing component, (2) a non-enhancing component, (3) a necrotic component and (4) a edema component.)

Integrative Cancer Research with Digital Pathology

High-resolution whole-slide microscopy

Visually, when scanning through the entire tumor volume, what proportion of the entire tumor would you estimate is enhancing. (Assuming that the entire abnormality may be comprised of: (1) an enhancing component, (2) a non-enhancing component, (3) a necrotic component and (4) a edema component.)

Integrative Analysis: OSU BISTI NIB Center
Big Data (2005)

Associate genotype with phenotype
Big science experiments on cancer, heart disease, pathogen host response

- Tissue specimen -- 1 cm³
- 0.1 μ resolution -- roughly 10^{15} bytes

Molecular data (spatial location) can add additional significant factor; e.g. 10^5

- Multispectral imaging, laser captured microdissection, Imaging Mass Spec, Multiplex QD
- Multiple tissue specimens; another factor of 10^3

Total: 10^{20} bytes -- 100 exabytes per big science experiment

Source - IEEE Big Data Conference 2015
# Big Data vs HPC

<table>
<thead>
<tr>
<th></th>
<th>Big Data</th>
<th>HPC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Applications</strong></td>
<td>Data analytics: Social networks, industry</td>
<td>Large-scale scientific simulation: government, industry</td>
</tr>
<tr>
<td><strong>Characterized by</strong></td>
<td>Typically, independent file operations, database queries</td>
<td>Typically map to 3-D grid to represent physical space</td>
</tr>
<tr>
<td><strong>Prevalent data abstractions</strong></td>
<td>Graphs (sparse), databases, text files</td>
<td>Arrays (dense and sparse), objects</td>
</tr>
<tr>
<td><strong>Programming Models</strong></td>
<td>Map-Reduce/HIVE/Giraph etc.</td>
<td>MPI/OpenMP/CUDA widely used</td>
</tr>
<tr>
<td><strong>Failure Model</strong></td>
<td>Assume failures common, need to be tolerated</td>
<td>Assume failures infrequent (spend $)</td>
</tr>
<tr>
<td><strong>System Cost</strong></td>
<td>Use the technology with the best price-performance ratio</td>
<td>Use the fastest possible processors/network</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Challenges (Exascale/Big Data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Energy budget limitation</td>
</tr>
<tr>
<td>• Interconnect tightly couple</td>
</tr>
<tr>
<td>• Memory, hierarchical</td>
</tr>
<tr>
<td>• Scalable system software</td>
</tr>
<tr>
<td>• Programming systems</td>
</tr>
<tr>
<td>• Data management</td>
</tr>
<tr>
<td>• Network, Workflow engine</td>
</tr>
<tr>
<td>• Exascale Algorithms</td>
</tr>
<tr>
<td>• Algorithm for recovery, fault tolerance, hard crashing</td>
</tr>
<tr>
<td>• Correctness, reproducitively</td>
</tr>
<tr>
<td>• Science productivity</td>
</tr>
<tr>
<td>• Real time simulation</td>
</tr>
<tr>
<td>• Energy Consumption</td>
</tr>
<tr>
<td>• Interconnect wide and open</td>
</tr>
<tr>
<td>• Memory, flat and big</td>
</tr>
<tr>
<td>• Scalable storage system</td>
</tr>
<tr>
<td>• Programming tools</td>
</tr>
<tr>
<td>• Data management</td>
</tr>
<tr>
<td>• Network, Workflow engine</td>
</tr>
<tr>
<td>• Exabyte Data Algorithms</td>
</tr>
<tr>
<td>• Algorithm for recovery, fault tolerance, soft landing</td>
</tr>
<tr>
<td>• Stochastic convergent, reproducitively</td>
</tr>
<tr>
<td>• Conclusive guidance and predictive conclusion</td>
</tr>
</tbody>
</table>
Challenges - Big Data/Exascale

Integration is challenging in both directions

Application Level
- Mahout, R and Applications
- Hive
- Pig
- Sqoop
- Flume
- Map-Reduce
- Storm
- Hbase BigTable (key-value store)
- HDFS (Hadoop File System)

Middleware & Management
- Zookeeper (coordination)
- VMs and Cloud Services (optional)
- Linux OS variant

System Software
- Ethernet Switches
- Local Node Storage
- Commodity X86 Racks

Cluster Hardware
- Data Analytics Software Stack
- Computational Science Software Stack

Software
- FORTRAN, C, C++ and IDEs
- Domain-specific Libraries
- MPI-OpenMP CUDA/OpenCL
- NA Libs
- Perf & Debug (e.g., PAPI)
- PFS (e.g., Lustre)
- Batch Scheduler
- System Monitoring

Big Data in Machine Learning – GPU acceleration

Source captured frin - Julie Bernauer – HPC Advisory Council Stanford Tutorial – 2017/02/07
Need of Parallel Computer

- Requirement of computational capacity depends on applications and formulations and what you want to achieve
- **Length Scale** (memory) - resolution of the dimension, e.g. number of grid points
- **Time Scale** (fast) - resolution of duration, e.g. number of time step

2D problem:
- grid points 100x100 = 10000 pts
- a vector of 10000 elements ~ 80 KB
- need 10 such vectors ~ 800KB
- Steady State in seconds

3D problem:
- grid points 10000x10000 x100 = 10e10 pts
- 10e10 unknowns ~ 80 GB
- need 10 such vectors ~ 800 GB MEMORY
- 100 years simulation !!

**NEED MULTIPLE WORKERS and MEMORY – PARALLEL COMPUTER**
Parallel Computing

Division of work into smaller tasks
Multiple computers work on smaller tasks simultaneously

>> Reduce Wall Clock Time <<
Issues of Parallel Computing

• Pros :
  – decrease wallclock time
  – deliver huge amount of memory
  – Allow realistic simulation

• Cons :
  – Difficult to construct
  – Efficient parallel algorithm may need some thoughts
  – Cost of program development

KEYS:
1) LOAD BALANCE - same amount of work for every processor
2) LOCALITY - minimize communications among processors
3) PORTABILITY - work well on different platforms of computers
4) SCALABILITY - can solve larger problem efficiently
Parallel Programming Example: Calculating Pi

- Use numerical integration to compute Pi
- Let \( f(x) = \frac{4}{1+x^2} \) then integrate \( f(x) \) from \( x = 0 \) to \( 1 \)
- Using the rectangle rule

\[
R_n(f) = h \sum_{i=1}^{n} f(x_i)
\]

where \( n = \) the number of intervals, \( h = 1/n \) is the rectangle width and \( x_i = h.(n-0.5) \) is the midpoint of each rectangle

\[\text{Pi} = \text{area under } f(x)\]
Pi Using Rectangles

- Method: Divide area under curve into rectangles and distribute the rectangles to the processors
- Suppose there are 3 processors, how should the distribution be done?
Parallel Performance Measure

• Using multiple processors you hope your program will go faster
• Observed Speedup using N processors to accomplish a task

\[
\text{Speedup} = \frac{T(1)}{T(N)} \quad \text{Time taken using 1 processor}
\]

\[
\frac{T(1)}{T(N)} \quad \text{Time taken using N processors}
\]

• To be fair, should use the “best” serial algorithm on 1 processor, not the parallel algorithm, simply restricted to 1 processor
• Linear speedup:
  – Two processors take 1/2 the time of 1 processor, so speedup =2
  – N processors take 1/N the time of 1 processor, so speedup =N
• Superlinear speedup
  – May be obtained occasionally, usually due to cache and memory improvements
Amdahl’s Law

• Maximum speedup is limited by the serial fraction of a program

• Serial code
  
  \[\text{parallelizable (90)} \quad \text{Serial (10)}\]

  – Time taken: 100

• Parallel code (using num procs P >> 10)

  \[\text{parallel (0)} \quad \text{serial(10)}\]

  – Time taken =10, maximum speedup=10, regardless of P
Parallel Computers (simple story)

Shared Memory Systems (SMP)
(Multicore Node)
(Thread-base, OpenMP, )

Distributed Memory Systems (MPP)
(IBM SP, Cray XT or PC Cluster)
(USE MESSAGE PASSING)

P/C P/C P/C

Intra-node, switch

shared memory

M M M

P/N P/N P/N

Communication Network inter-node
Modern Supercomputers

Commodity plus Accelerator Today

**Commodity**
- Intel Xeon
- 8 cores
- 3 GHz
- 8*4 ops/cycle
- 96 Gflop/s (DP)

**Accelerator (GPU)**
- Nvidia K20X “Kepler”
- 2688 “Cuda cores”
- .732 GHz
- 2688*2/3 ops/cycle
- 1.31 Tflop/s (DP)

**Accelerator (Intel PHI)**
- 192 Cuda cores/SMX
- 2688 “Cuda cores”

Interconnect
- PCI-X 16 lane
- 64 Gb/s (8 GB/s)
- 1 GW/s

From ICL Dr. Jack Dongarra: icl.cs.utk.edu
GPU architecture:

Streaming Multiprocessors (SMs) - 32 cores

Reference: http://nvidia.com
GPU programming model:

- GPU accelerator is called device, CPU is host.
- GPU code (kernel) is launched and executed on the device by several threads.
- Threads grouped into thread blocks.
- Program code is written from single thread's point of view.
  - Each thread can diverge and execute a unique code path (can cause performance issues)
- Compute Unified Device Architecture (CUDA)
Introduction to CUDA:

- Compute Unified Device Architecture
- CUDA is a C/C++ language extension for GPU programming.
  - PGI has developed similar FORTRAN 2003 extension.
- Two APIs: Runtime and Driver
CUDA applications:

- Computational Geoscience
- Computational Chemistry
- Computational Medicine
- Computational Modeling
- Computational Science
- Computational Biology
- Computational Finance
- Image Processing
Good Practices

Use existing libraries
Understand the issues
Does it worth it to start from scratch
Ask the experts
1. Write the program, or build it from previous codes, etc.
2. Debug your code (with optimization switches off)
3. Ensure mathematical correctness of the program!
4. Profile your code – determine where most of the computing time is spent
5. Optimize the algorithm, the data mapping, the communication, the I/O
6. Try out different combinations of compiler flags and/or compiler directives
7. Profile your code again
8. Re-examine blocks of code that consume the most execution time
9. Repeatedly apply various optimizations to such blocks
10. Rerun optimized code, compare performance, and start again until “satisfied”.
Final thoughts: Strategies for Improved Performance

• Improving performance is a complex task, and the amount of time and effort put into it might not always be worth it.

• A certain trade-off must be reached between the developmental time and the "final" production run time.

• If you need to work on a previously existing code, then take the time to learn the details of its logic (if possible). Sometimes you might be better off rewriting the whole code directly in parallel!

• If you write the program from scratch, take some time to think about the different performance issues presented here and/or elsewhere.

• Examine benchmark results and know the limits of the computing platform

Finally: What else can be done?

– Practice, try new approaches, innovate, ask others
– Remember to concentrate only on subroutines worth improving
– Rethink the whole algorithm from scratch !?
– Remember to re-check the results for “correctness” (whenever possible!)
– Change parallel method (?), or change parallel machine (?)
– (ask someone else to do the calculations! ;-)}
Mapping Problem: Decomposition

- Each processor should have a similar amount of work
- Expensive communications should be minimized.
- Communications should be:
  - eliminated where feasible
  - localized otherwise (i.e. communicate between close CPU neighbors) (not crucial anymore)
- Concurrency should be maximized
- NOTE: finding the best mapping is an NP-complete problem! :-(

1D Decomposition

2D Decomposition

2D Block Cyclic
Load Balancing

- **Static**
  - Data or tasks are partitioned initially among the existing node processors
  - Problem: finding a good initial mapping of data or tasks to the processors

- **Dynamic**
  - Assumes there is a pool of tasks which can be selected and distributed at runtime (e.g. a task queue or bag_of_tasks)
  - Next available task is assigned to a free processor
  - Or, it implies that the data can be redistributed appropriately during execution of program
  - Problem: Synchronization issues
Communication Characteristics

- Relatively slow communication vs. computation
  - Peak bandwidths: ~1 MB/sec with ethernet connections
  - 12.5 MB/sec with a 100 Mbit/sec switch network
  - 150 MB/sec on the SP2
  - 9.6 GB/sec on the Cray XT5 between nodes
  - Implies advantage of using either coarse- or medium-grained parallelism
- The bigger communication cost is in the "startup" or latency
- Overhead - 40 usec (software) latency on the SP2
  - Sending separate 1-byte messages --> 1s/40us = 25 KBytes/sec !!
  - Better sending few large messages rather than many small ones
  - Cray XT5 - latency: a few us
- Bottom line: try to minimize the ratio of
  - (# messages) / (# computations)
Communication Issues

• Contentions, or traffic jams
  – Have good distribution of messages. Circular or round-robin methods in one or two dimensions are fairly efficient for certain problems.
  – Avoid as much as possible the use of indirect addressing.
  – Use threads on multicore

• Ready mode in MPI or post receive before send
  – use MPI_Rsend when you are *sure* that a matching receive (MPI_Recv) has been posted appropriately
  – this allows faster transfer protocols
  – HOWEVER! behavior is undefined if receive was not posted in time!
  – Post receive before send on Cray

• Mask communication with computation
  – Use asynchronous mode,
  – Avoid barrier
I/O and Parallel I/O

- I/O can be a serious bottleneck for certain applications. The time to read/write data to disks could be an issue. But sometimes the shear size of the data file is a problem.

- Parallel I/O systems allow (in theory) the efficient manipulation of huge files.

- Unfortunately, parallel I/O is only available on some architectures, and software is not always good. (MPI-2 has parallel MPI-IO on ROMIO implementation)

- They are restricted to few (around 4 or so) parallel disk drives, through designated I/O nodes.

- On the IBM with GPFS

- Lustre on the ACF System

- One single files vs file/process

- Using local /tmp for input output

- Progress is still needed in this area!
Improving performance is a complex task, and the amount of time and effort put into it might not always be worth it.

A certain trade-off must be reached between the developmental time and the "final" production run time.

If you need to work on a previously existing code, then take the time to learn the details of its logic (if possible). Sometimes you might be better off rewriting the whole code directly in parallel!

If you write the program from scratch, take some time to think about the different performance issues that we have been presenting here.

Examine benchmark results and know the limit of the computing platform.

Profilers "prof" give information on:
- how much time (seconds) is spent in each subroutine
- what percentage of time each subroutine is consuming
- the cumulative time
- the # of calls to subroutines made
- the time (msecs) per call
- Use available system tools
Performance Tuning Process

- 1. Debug your code (with optimization switches off)
- 2. Ensure mathematical correctness of the program!
- 3. Profile your code
- 4. Optimize the algorithm
- 5. Compile with optimization switches on
- 6. Profile your code
- 7. Examine blocks of code that consume the most execution time
- 8.Repeatedly apply various optimizations to such blocks
- 9. Ensure again the numerical correctness of the program!

• Finally: What else can be done?
  - Practice, try new approaches, innovate, ask others
  - Concentrate only on subroutines worth improving
  - Rethink the whole algorithm from scratch !?
  - Re-check the results for correctness (whenever possible!)
  - Change parallel method (?)
  - Change parallel machine (?)
  - (ask someone else to do it! ;-)
Writing Parallel Programs

- Use prewritten programs
  - There are parallel database codes, genetic algorithms, neural networks, linear algebra, etc available

- Writing code to take advantage of parallel libraries
  - Use libraries like ScaLAPACK (Scalable Linear Algebra Package), and other optimized parallel libraries in your code
  - Usually much faster and more robust than code you could easily write

- Writing your own code from scratch
  - The hardest choice… but used by many because of its flexibility
The End

Quote: “I think there is a world market for maybe five computers”
Thomas Watson, chairman of IBM, 1943