High-Performance Computing: Past, Present and Future

Carlos P Sosa
IBM & Biomedical Informatics and Computational Biology
University of Minnesota Rochester
Past
- High-Performance Computing: A Historical Perspective
- Big Irons
- Minisupercomputers
- Challenges and Limitations
- Large power consumption
- Hardware features
- Areas of applications

Present and Future
- GPGPUs
  - Hardware features
  - Programming paradigms
  - Applications
- Cloud Computing

Summary
High-performance computing (HPC) uses supercomputers to solve advanced computation problems.

Today, computer systems approaching the teraflops-region are counted as HPC-computers.

Measure of a computer's processor speed. This speed can be expressed as a trillion floating point operations per second, 10^{12} floating-point operations per second.

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Symbol</th>
<th>$1000^m$</th>
<th>$10^n$</th>
<th>Decimal</th>
<th>Short scale</th>
<th>Long scale</th>
<th>Since[n 1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>yotta</td>
<td>Y</td>
<td>$1000^8$</td>
<td>$10^{24}$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Septillion</td>
<td>Quadrillion</td>
<td>1991</td>
</tr>
<tr>
<td>zetta</td>
<td>Z</td>
<td>$1000^7$</td>
<td>$10^{21}$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Sextillion</td>
<td>Trilliard</td>
<td>1991</td>
</tr>
<tr>
<td>exa</td>
<td>E</td>
<td>$1000^6$</td>
<td>$10^{18}$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Quintillion</td>
<td>Trillion</td>
<td>1975</td>
</tr>
<tr>
<td>peta</td>
<td>P</td>
<td>$1000^5$</td>
<td>$10^{15}$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Quadrillion</td>
<td>Billion</td>
<td>1975</td>
</tr>
<tr>
<td>tera</td>
<td>T</td>
<td>$1000^4$</td>
<td>$10^{12}$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Trillion</td>
<td>Billion</td>
<td>1960</td>
</tr>
<tr>
<td>giga</td>
<td>G</td>
<td>$1000^3$</td>
<td>$10^9$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Billion</td>
<td>Milliard</td>
<td>1960</td>
</tr>
<tr>
<td>mega</td>
<td>M</td>
<td>$1000^2$</td>
<td>$10^6$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Million</td>
<td>Milliard</td>
<td>1960</td>
</tr>
<tr>
<td>kilo</td>
<td>k</td>
<td>$1000^1$</td>
<td>$10^3$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Million</td>
<td>Million</td>
<td>1795</td>
</tr>
<tr>
<td>hecto</td>
<td>h</td>
<td>$1000^{2/3}$</td>
<td>$10^2$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Million</td>
<td>Million</td>
<td>1795</td>
</tr>
<tr>
<td>deca</td>
<td>da</td>
<td>$1000^{1/3}$</td>
<td>$10^1$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Million</td>
<td>Million</td>
<td>1795</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000^0</td>
<td>$10^0$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Million</td>
<td>Million</td>
<td>1795</td>
</tr>
<tr>
<td>deci</td>
<td>d</td>
<td>$1000^{-1/3}$</td>
<td>$10^{-1}$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Million</td>
<td>Million</td>
<td>1795</td>
</tr>
<tr>
<td>centi</td>
<td>c</td>
<td>$1000^{-2/3}$</td>
<td>$10^{-2}$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Million</td>
<td>Million</td>
<td>1795</td>
</tr>
<tr>
<td>milli</td>
<td>m</td>
<td>$1000^{-1}$</td>
<td>$10^{-3}$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Million</td>
<td>Million</td>
<td>1795</td>
</tr>
<tr>
<td>micro</td>
<td>μ</td>
<td>$1000^{-2}$</td>
<td>$10^{-6}$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Million</td>
<td>Million</td>
<td>1960[n 2]</td>
</tr>
<tr>
<td>nano</td>
<td>n</td>
<td>$1000^{-3}$</td>
<td>$10^{-9}$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Million</td>
<td>Million</td>
<td>1960</td>
</tr>
<tr>
<td>pico</td>
<td>p</td>
<td>$1000^{-4}$</td>
<td>$10^{-12}$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Million</td>
<td>Million</td>
<td>1960</td>
</tr>
<tr>
<td>femto</td>
<td>f</td>
<td>$1000^{-5}$</td>
<td>$10^{-15}$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Million</td>
<td>Million</td>
<td>1960</td>
</tr>
<tr>
<td>atto</td>
<td>a</td>
<td>$1000^{-6}$</td>
<td>$10^{-18}$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Million</td>
<td>Million</td>
<td>1960</td>
</tr>
<tr>
<td>zepto</td>
<td>z</td>
<td>$1000^{-7}$</td>
<td>$10^{-21}$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Million</td>
<td>Million</td>
<td>1960</td>
</tr>
<tr>
<td>yocto</td>
<td>y</td>
<td>$1000^{-8}$</td>
<td>$10^{-24}$</td>
<td>1,000,000,000,000,000,000,000,000</td>
<td>Million</td>
<td>Million</td>
<td>1960</td>
</tr>
</tbody>
</table>

1. * The metric system was introduced in 1795 with six prefixes. The other dates relate to recognition by a resolution of the CGPM.  
2. * The 1948 recognition of the micron by the CGPM was abrogated in 1967.
High-Performance Computers were introduced in the 1960s and were designed primarily by Seymour Cray at Control Data Corporation (CDC)

- Led the market into the 1970s
- Founded Cray Research
- Big irons dominated the market (1985-1990)
- 1980s the decade of the minicomputer
- Mid-1990s "supercomputer market crash"
Big Irons

1985 Cray-2

IBM 3090

Source: http://www.mrynet.com/cray/docs.html

Source: http://www.bobndenise.com/computers/computer.htm
### Minisupercomputers 1980's

**CONVEX COMPUTER CORP.**

<table>
<thead>
<tr>
<th>NAME</th>
<th>C3800</th>
</tr>
</thead>
<tbody>
<tr>
<td>MANUFACTURER</td>
<td>Convex Computer Corp.</td>
</tr>
<tr>
<td>TYPE</td>
<td>Professional Computer</td>
</tr>
<tr>
<td>ORIGIN</td>
<td>U.S.A.</td>
</tr>
<tr>
<td>YEAR</td>
<td>1991</td>
</tr>
<tr>
<td>BUILT IN LANGUAGE</td>
<td>None</td>
</tr>
<tr>
<td>KEYBOARD</td>
<td>Video terminal</td>
</tr>
<tr>
<td>CPU</td>
<td>2 to 8 processors</td>
</tr>
<tr>
<td>SPEED</td>
<td>240 Mflops to 2 Gflops</td>
</tr>
<tr>
<td>RAM</td>
<td>512 MB to 4 GB</td>
</tr>
<tr>
<td>TEXT MODES</td>
<td>Depending on the video terminal used</td>
</tr>
<tr>
<td>GRAPHIC MODES</td>
<td>Ditto</td>
</tr>
<tr>
<td>SIZE / WEIGHT</td>
<td>178 (H) x 201-328 (W) x 224 (D) cm / 1270 to 2631 kg</td>
</tr>
<tr>
<td>I/O PORTS</td>
<td>At least 8 x 80 Mbytes/sec PBUS slots, Integrated Disc Channel (IDC), Tape Library Interface (TLI)</td>
</tr>
<tr>
<td>BUILT IN MEDIA</td>
<td>From 34 GB hard disk</td>
</tr>
<tr>
<td>OS</td>
<td>Convex OS</td>
</tr>
<tr>
<td>POWER SUPPLY</td>
<td>Power consumption from 19.8 to 57.2 KVA - Forced air cooling</td>
</tr>
<tr>
<td>PERIPHERALS</td>
<td>Unknown</td>
</tr>
<tr>
<td>PRICE</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

Technological challenges

- The point to which we can shrink transistors has an absolute limit
- The shrinking of transistors yield difficult side effects (Electro-Magnetic Interference)
- Power leakage

Moore’s Law predicts that the number of transistors that can be economically placed on an integrated circuit will double about every two years.

But by 2020, Moore’s Law is expected to hit a brick wall, as manufacturing costs rise and transistors shrink beyond the reach of the laws of classical physics.

A supercomputer consumes large amounts of electrical power:

- Almost all of which is converted into heat, requiring cooling
- Tianhe-1A consumes 4.04 Megawatts of electricity
- The cost to power and cool the system is usually one of the factors that limit the scalability of the system
- 4MW at $0.10/KWh is $400 an hour or about $3.5 million per year

Source: http://en.wikipedia.org/wiki/Supercomputer
Commodity computing

Large-scale machines could be achieved using individual CPUs networked, or clustered to function together as a single unit

Massively parallel processing (MPP) systems

From Kilobytes to Petabytes in 50 Years: http://www.eurekalert.org/features/doe/2002-03/dlnl-fkt062102.php
Supercomputer Peak Performance

Biomedical Informatics & Computational Biology

http://www.reed-electronics.com/electronicnews/article/CA508575.html?industryid=21365
From Kilobytes to Petabytes in 50 Years: http://www.eurekalert.org/features/doe/2002-03/dlnl-fkt062102.php
Grand challenge* problems is a key part of high performance computing applications.

Grand challenges are fundamental problems in science and engineering with broad economic and scientific impact, and whose solution can be advanced by applying high performance computing techniques and resources.
Different from the Rest

Source: Pete Beckam, Director, ACLF, Argonne National Lab.
Source: Pete Beckam, Director, ACLF, Argonne National Lab.
- Limits of physical size (floor space)
- Power consumption
- Cooling needed to house and run the aggregated equipment
Design Considerations

- Widening gap between processor and DRAM clock rates
- Excessive heat generated by dense packaging and high switching frequency
- Disparity between processor clock rate and immediate vicinity peripheral devices (memory, I/O buses, etc.)
- Network performance

The speed of the processor is traded in favor of dense packaging and low power consumption per processor.
Most Power, Space, and Cooling efficient Supercomputer

(Published specs per peak performance)
Operating System in HPC

Source: http://en.wikipedia.org/wiki/Supercomputer
The first programmable electronic computer (ENIAC) was first used by John von Neumann at LANL to do calculations for the Manhattan Project.

The first high-level programming language FORTRAN was developed by John Backus in 1954 for the IBM 704 which was the first mass-produced computer with floating-point and also used by LLNL.

IBM’s System/360 was the first computer designed for both commercial and scientific applications, including a floating-point unit.

VM/370 (grandfather of z/VM and really all hypervisors) was developed in the early 70’s by the Cambridge Scientific Center and was a mainstay at High Energy Physics labs like CERN and SLAC because of it’s pioneering open source model.

In 1986, Dennis Jennings who was in charge of Supercomputing at NSF chose DARPA’s TCP/IP protocols for NSFNET and in 1988 NSFNET was opened to commercial traffic and the Internet was born.

In 1989, Tim Berners-Lee at CERN invented the World Wide Web and in 1993 NCSA released the popular Mosaic web browser.

An IBM SP2 supercomputer was used at the 1996 Atlanta Olympics for large scale web serving using a suite of software technologies that ultimately became WebSphere.

Loadleveler Interactive Session Services used to load balance login services at SP2 supercomputer sites became SWG’s eNetwork Dispatcher product and Cisco’s first network appliance product.

Intel’s use of thousands of SP2 nodes for x86 processor development drove the development of numerous improvements in NFS, scalable NFS, SPECSFS, and Auspex which invented the NAS product category.

Ian Foster and his colleagues at Argonne National Lab in the 90’s developed the concept known as Grid Computing to solve problems in High Energy Physics and in collaboration with Jeff Nick at IBM developed Web Services.

HPC drove the use and adoption of Infiniband and RDMA.

Blue Gene was the first large-scale system to use energy-efficient processors.

CERN was the first IBM StorageTank customer. StorageTank introduced the concept of policy driven tiered storage pools which has been integrated into GPFS and is now a common feature of most commercial storage products.

Perl was developed by Larry Wall while working at NASA as a systems administrator.

Message: Many major technical products that are commonplace were once state-of-the-art for supercomputing – hypervisors, FORTRAN, TCP/IP, the Internet, WWW, web browser, WebSphere (middleware), NAS, IB, RDMA, Perl..
**Areas of Application**

**Biomedical Informatics & Computational Biology**

- **Improve understanding** - significantly larger scale, more complex and higher resolution models; new science applications
- **Multiscale and multiphysics** - From atoms to mega-structures; coupled applications
- **Shorter time to solution** - Answers from months to minutes

---

**Physics – Materials Science**
- Molecular Dynamics

**Geophysical Data Processing**
- Upstream Petroleum

**Biological Modeling**
- Brain Science

**Life Sciences: In-Silico**
- Trials, Drug Discovery

**Computational Fluid Dynamics**

**Geophysical Data Processing**
- Streaming Data Analysis

**Environment and Climate Modeling**

**Life Sciences: Sequencing**

---

**Financial Modeling**
The European Centre for Medium-Range Weather Forecasts (ECMWF, the Centre) is an independent intergovernmental organization supported by 32 States: Belgium, Denmark, Germany, Greece, Spain, France, Ireland, Italy, Luxembourg, the Netherlands, Norway, Austria, Portugal, Switzerland, Finland, Sweden, Turkey, United Kingdom.

Forschungszentrum Jülich pursues cutting-edge interdisciplinary research on solving the grand challenges facing society in the fields of health, energy and the environment, and also information technologies.
<table>
<thead>
<tr>
<th>Year</th>
<th>Gigaflops</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>1993</td>
<td>236</td>
<td>Numerical Wind Tunnel</td>
</tr>
<tr>
<td>1996</td>
<td>614</td>
<td>CP-PACS</td>
</tr>
<tr>
<td>1997</td>
<td>1830</td>
<td>ASCI Red</td>
</tr>
<tr>
<td>1999</td>
<td>3207</td>
<td>ASCI Red</td>
</tr>
<tr>
<td>2000</td>
<td>12,288</td>
<td>ASCI White</td>
</tr>
<tr>
<td>2002</td>
<td>40,960</td>
<td>Earth-Simulator</td>
</tr>
<tr>
<td>2004</td>
<td>91,750</td>
<td>IBM BG/L</td>
</tr>
<tr>
<td>2005</td>
<td>367,000</td>
<td>LLNL BG/L</td>
</tr>
<tr>
<td>2007</td>
<td>596,378</td>
<td>LLNL BG/L</td>
</tr>
<tr>
<td>2008</td>
<td>1,456,704</td>
<td>Los Alamos Roadrunner</td>
</tr>
<tr>
<td>2009</td>
<td>2,331,000</td>
<td>Oak RidgeJaguar (Cray)</td>
</tr>
<tr>
<td>2010</td>
<td>4,701,000</td>
<td>Tianhe-1A</td>
</tr>
</tbody>
</table>
Present and Future
Next Milestone

- Graphics processing units (GPUs)
  - Offload graphics-related rendering operations in a computer
- Offload general purpose (GP) application workloads and significantly speed up part of the computation
  - Becoming mainstream
Combining CPU and GPU for computations

CPU + GPU Co-Processing

4 cores
48 GigaFlops (DP)

448 cores
GPU
515 GigaFlops (DP)

Source: http://www.redbooks.ibm.com/abstracts/sg247629.html
NVIDIA:
- Tesla products (latest architecture is Fermi)
- Up to 512 CUDA cores
- SP 1.03 Tflops, DP 515 Gflops
- Peak memory bandwidth 148 GB/s

AMD/ATI:
- FireStream product range
- 800 Stream cores
- SP 1.2 Tflops, DP 240 Gflops
- Peak memory bandwidth 102 GB/s

Intel Larrabee:
- “On May 25, 2010 the Technology@Intel blog announced that Larrabee would not be released as a GPU, but instead would be released as a product for High Performance Computing competing with the Nvidia Tesla”
Fermi features several major innovations:

- 3 billion transistors
- Up to 512 CUDA cores
- Parallel DataCache technology
- Added L1 and L2 Caches
- Improved GigaThread Hardware Thread Scheduler Engine
- ECC on all Internal and External Memories
- High Speed GDDR5 Memory Interface

*http://www.nvidia.com/page/home.html*
M (Module) solutions differ from C (Card) solutions in several ways:

- M’s are passive; C’s are actively cooled
- M’s don’t have I/O connectors, C’s do (DVI)
- C’s have a full OpenGL accelerated engine, M’s do not.
- M’s for data center products (like idpx and BladeCenter)
Key Research Applications Available on GPUs

Molecular Dynamics / Quantum Chemistry

Computational Fluid Dynamics

Astrophysics

Weather & Climate Modeling

Many More

AMBER
ABINIT
DL_POLY
GROMACS
LAMMPS
MADNESS
NAMD
Q-Chem
TeraChem

OpenCurrent
BAE Systems
ANDSolver
Euler Solvers
Lattice
Boltzman
Navier Stokes

N-body
Chimera
GADGET2
Many published papers

ASUCA (Japan)
CO2 Modeling (Japan)
HOMME
Tsunami modeling
NOAA NIM WRF

• Materials Science
• DCA++
• gWL-LSMS
• Combustion
• S3D
• Lattice QCD
• Chroma (QUDA)

http://www.nvidia.com/object/tesla_bio_workbench.html
GPU Parallel Computing Developer Eco-System

Biomedical Informatics & Computational Biology

Numerical Packages
- MATLAB
- Mathematica
- NI LabView
- pyCUDA

Debuggers & Profilers
- cuda-gdb
- NV Visual Profiler
- “Nexus” VS 2008
- Allinea
- TotalView

GPU Compilers
- C
- C++
- Fortran
- OpenCL
- DirectCompute
- Java
- Python

Parallelizing Compilers
- PGI Accelerator
- CAPS
- HMPP
- mCUDA
- OpenMP

Libraries
- CUBLAS
- CUFFT
- CULA
- NPP
- Video Imaging

http://www.nvidia.com/page/home.html
Cluster Tools Integration

- Platform Computing:
  - Cluster Manager CUDA kit
- Penguin Computing: Scyld
- Cluster Corp: Rocks Roll
- Altair: PBS Works
<table>
<thead>
<tr>
<th></th>
<th>Tesla M1060</th>
<th>Tesla M2050</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong># cores</strong></td>
<td>240</td>
<td>448</td>
</tr>
<tr>
<td><strong>Frequency of processor cores</strong></td>
<td>1.3 GHz</td>
<td>1.15 GHz</td>
</tr>
<tr>
<td><strong>Double Precision floating point performance (peak)</strong></td>
<td>78 Gflops</td>
<td>515 Gflops</td>
</tr>
<tr>
<td><strong>Single Precision floating point performance (peak)</strong></td>
<td>933 Gflops</td>
<td>1.03 Tflops</td>
</tr>
<tr>
<td><strong>Total Dedicated Memory</strong>*</td>
<td>4GB DDR3</td>
<td>3GB GDDR5</td>
</tr>
<tr>
<td><strong>Memory Clock</strong></td>
<td>1.6 GHz</td>
<td>3000 GHz</td>
</tr>
<tr>
<td><strong>Memory Speed</strong></td>
<td>800 MHz</td>
<td>1.55 GHz</td>
</tr>
<tr>
<td><strong>Memory Interface</strong></td>
<td>512-bit</td>
<td>384-bit</td>
</tr>
<tr>
<td><strong>Memory Bandwidth</strong></td>
<td>102 GB/sec</td>
<td>148 GB/sec</td>
</tr>
<tr>
<td><strong>Power Consumption</strong></td>
<td>190W</td>
<td>225W</td>
</tr>
</tbody>
</table>
Performance Summary

Biomedical Informatics & Computational Biology

Performance Summary

MIDG: Discontinuous Galerkin Solvers for PDEs

AMBER Molecular Dynamics (Mixed Precision)

Lock Exchange Problem OpenCurrent

OpenEye ROCS Virtual Drug Screening

Radix Sort CUDA SDK

Standard FFT Library: cuFFT 3.1-Pre-release

BICB
Biomedical Informatics & Computational Biology

cuFFT 3.1-pre-release: NVIDIA Tesla C1060 GPU and Tesla C2050 (Fermi)
MKL 10.1r1: Quad-Core Intel Core i7 (Nehalem) 3.2GHz

Standard BLAS Library: cuBLAS 3.1-Pre-release

cuBLAS: CUDA 3.1-pre-release, Tesla C1060 and Tesla C2050

MKL 10.1r1: Intel Core2 Extreme, 3.00GHz

Consistent Speedups over 3 Generations

Molecular orbital calculations in VMD

Speedup

Short-Range Electrostatics Kernels in VMD

Speedup

http://www.ks.uiuc.edu/Research/vmd/

http://www.nvidia.com/page/home.html
CUDA vs OpenCL

- Tests performed on Fermi C2050
- http://blog.accelereyes.com/blog/2010/05/10/nvidia-fermi-cuda-and-opencl/
- Ability to feed data to the GPU fast enough could be a limiting factor
- Writing results to disk before re-computing may also take place

- From the above, bottlenecks for applications could be:
  - Bus bandwidth/latency between GPU and CPU
  - Capacity/speed of server host memory
  - Speed of local hard-drives
  - Bandwidth/latency of network connectivity
IBM iDataPlex (x86)

IBM Biomedical Informatics & Computational Biology

- iDataPlex is:
  - Optimized both mechanically as a half-depth server solution and component-wise for maximum power and cooling efficiency
  - Designed to maximize utilization of Data Center floor space, power and cooling infrastructure with Industry-standards based server platform.
  - Easy-to-maintain solution with individually serviceable servers, front access hard drives/cabling,
  - Configurable for customer-specific compute, storage, or I/O needs and delivered pre-configured for rapid deployment
  - Common tools across the System x portfolio for management at the node, rack or Data Center level
dx360 M3 Refresh - Server GPU Configuration

- 4-2.5” SS SAS 6Gbps (or SATA, or 3.5”, or SSD…)
- NVIDIA Tesla M2050 #1 (or NVIDIA Tesla M1060, or FX3800, or Fusion IO, or)
- NVIDIA Tesla M2050 #2 (or NVIDIA Quadro FX3800, or Fusion IO, or…)
- Infiniband DDR (or QDR, or 10GbE…)

CPU + GPU Co-Processing

- CPU: 48 GigaFlops (DP)
- GPU: 515 GigaFlops (DP)
dx360 M3 Refresh - New 3-slot Riser and I/O Tray

BICB
Biomedical Informatics & Computational Biology

2 PCIe x16 (1 per side)

PCIE x8

GPU 1

GPU 2

HBA
# IBM iDataPlex with GPGPU Machine

## Biomedical Informatics & Computational Biology

<table>
<thead>
<tr>
<th>Feature</th>
<th>dx360 M2 server</th>
<th>dx360 M3 server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cores per processor</td>
<td>2 or 4</td>
<td>2, 4, or 6</td>
</tr>
<tr>
<td>Memory speed</td>
<td>800, 1066, or 1333 MHz</td>
<td>800, 1066, or 1333 MHz</td>
</tr>
<tr>
<td>DIMM sockets</td>
<td>16 DIMM sockets</td>
<td>16 DIMM sockets</td>
</tr>
<tr>
<td>Maximum memory</td>
<td>128 GB DDR3</td>
<td>128 GB DDR3</td>
</tr>
<tr>
<td>PCIe</td>
<td>x16 (Gen 2)</td>
<td>x16 (Gen 2)</td>
</tr>
<tr>
<td>Graphics card support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GPGPU support</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Supported in 2U chassis</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Supported in 3U chassis</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SATA II support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SAS support</td>
<td>Yes - 3Gb&lt;sup&gt;a&lt;/sup&gt;</td>
<td>Yes - 6Gb&lt;sup&gt;a&lt;/sup&gt;</td>
</tr>
<tr>
<td>SSD support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Maximum internal storage per server</td>
<td>24 TB (using 3U chassis)</td>
<td>24 TB (using 3U chassis)</td>
</tr>
<tr>
<td>3.5-inch drive support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2.5-inch drive support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

---

*a. Requires an optional SAS controller card*
Figure 1. NAMD 2.7 runs using the apoA1 benchmark

Figure 2. AMBER 11 runs using the jac benchmark
#include "../common/book.h"

int main( void ) {
    printf ( "Hello, World:\n" );
    return 0;
}

J. Sanders and E. Kandrot, CUDA by Example
Host versus Device

CPU+ Memory

GPU+ Memory

Host

Device
A function that executes on the device is typically called *kernel*.

J. Sanders and E. Kandrot, *CUDA by Example*
#include "../common/book.h"

__global__ void kernel( void ) {
}

int main( void ) {
    kernel<<<1,1>>>();
    printf( "Hello, World!\n" );
    return 0;
}

J. Sanders and E. Kandrot, CUDA by Example
#include "../common/book.h"

__global__ void add( int a, int b, int *c ) {
    *c = a + b;
}

int main( void ) {
    int c;
    int *dev_c;
    HANDLE_ERROR( cudaMalloc( (void**)&dev_c, sizeof(int) ) );

    add<<<1,1>>>( 2, 7, dev_c );
    HANDLE_ERROR( cudaMemcpy( &c, dev_c, sizeof(int), cudaMemcpyDeviceToHost ) );

    printf( "2 + 7 = %d\n", c );
    HANDLE_ERROR( cudaFree( dev_c ) );

    return 0;
}
- We pass parameters to a kernel as we would with any C function.
- We need to allocate memory to do anything useful on a device, such as return values to the host.

J. Sanders and E. Kandrot, CUDA by Example
cudaError_t cudaMalloc (void ** devPtr, size_t size)

Allocates size bytes of linear memory on the device and returns in *devPtr a pointer to the allocated memory. The allocated memory is suitably aligned for any kind of variable. The memory is not cleared. cudaMalloc() returns cudaErrorMemoryAllocation in case of failure.

Parameters:
- devPtr - Pointer to allocated device memory
- size - Requested allocation size in bytes

Returns:
- cudaSuccess, cudaErrorMemoryAllocation
Utility macro as part of the book

- It detects that the call has returned an error, prints the associated error message, and exits the application with an EXIT_FAILURE code

```c
static void HandleError( cudaError_t err,
                      const char *file,
                      int line ) {
    if (err != cudaSuccess) {
        printf( "%s in %s at line %d\n",
                cudaGetErrorString( err ),
                file, line );
        exit( EXIT_FAILURE );
    }
}
#define HANDLE_ERROR( err ) (HandleError( err, __FILE__,
                                          __LINE__ ))
```

J. Sanders and E. Kandrot, CUDA by Example
Simplicity and Power of CUDA?

- Ability to blur the line between *host* and *device*
- It is responsibility of the programmer not to dereference the pointer returned by `cudaMalloc()` by the code that executes on the *host*
- *Host* code may pass the pointer around, perform arithmetic on it, or even cast it to a different type
- It CANNOT be used to read or write from memory

J. Sanders and E. Kandrot, CUDA by Example
Restrictions on the Usage of Device Pointers

- You can pass pointers allocated with `cudaMalloc()` to functions that execute on the **device**
- You can use pointers allocated with `cudaMalloc()` to read or write memory from codes that execute on the **device**
- You can pass pointers allocated with `cudaMalloc()` to functions that execute on the **host**
- You **CANNOT** use pointers allocated with `cudaMalloc()` to read or write memory from code that executes on the **host**

J. Sanders and E. Kandrot, CUDA by Example
cudaFree ()

Frees the memory space pointed to by devPtr, which must have been returned by a previous call to cudaMalloc() or cudaMallocPitch(). Otherwise, or if cudaFree(devPtr) has already been called before, an error is returned. If devPtr is 0, no operation is performed. cudaFree() returns cudaErrorInvalidDevicePointer in case of failure.

Parameters:

- devPtr - Device pointer to memory to free

Returns:

- cudaSuccess, cudaErrorInvalidDevicePointer, cudaErrorInitializationError

Note:

Note that this function may also return error codes from previous, asynchronous launches.
Two Methods for Accessing Device Memory

- By using device pointers from within the `device` code
- By using calls to `cudaMemcpy()`
cudaMemcpy() takes various parameters and returns a cudaError_t status code.

### Parameters:
- **dst**: Destination memory address
- **src**: Source memory address
- **count**: Size in bytes to copy
- **kind**: Type of transfer (cudaMemcpyHostToHost, cudaMemcpyHostToDevice, cudaMemcpyDeviceToHost, or cudaMemcpyDeviceToDevice)

### Returns:
- cudaSuccess
- cudaErrorInvalidValue
- cudaErrorInvalidDevicePointer
- cudaErrorInvalidMemcpyDirection

### Note:
- Calling cudaMemcpy() with dst and src pointers that do not match the direction of the copy results in an undefined behavior.
- Note that this function may also return error codes from previous, asynchronous launches.

- What are the capabilities of the device?
- What about when there is more than one CUDA-capable device per computer?

```c
cudaGetDeviceCount()
```

J. Sanders and E. Kandrot, CUDA by Example
- Iterate through the devices and query relevant information about each
- This information is stored in a structure of type cudaDeviceProp

J. Sanders and E. Kandrot, CUDA by Example
#include "../common/book.h"

int main( void ) {
    cudaDeviceProp prop;

    int count;
    HANDLE_ERROR( cudaGetDeviceCount( &count ) );
    for (int i=0; i< count; i++) {
        HANDLE_ERROR( cudaGetDeviceProperties( &prop, i ) );

        // Do something with our device’s properties
    }
}

J. Sanders and E. Kandrot, CUDA by Example
GPGPU an alternative for High-Performance Computing
Enables to do general purposes computations on a GPU
CUDA C is standard C with some ornamentation to allow us to specify which code should run on the device and which should run on the host
By adding the keyword __global__ before a function indicates to the compiler that we intend to run the function on the GPU
CUDA has: cudaMalloc (), cudaMemcpy (), and cudaFree ()
Cloud Computing as an Appliance

High Voltage Power Grid

Electricity is virtualized
Cloud Computing

Key ideas is to provide computing, storage and software as a service

“... a pay-per-use model for enabling available convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction...”

Technologies Behind Cloud Computing

- Utility and Grid Computing
- SOA
- Web 2.0
- Web Services
- Mashups
- Hardware virtualization
- Multi-core cips
- Autonomic Computing
- Data Center Automation

Buyya, Broberg, and Goscinski, Cloud Computing: Principles and Paradigms
Comparisons

- Autonomic Computing
- Client - server model
- Grid Computing
- Mainframe Computer
- Utility Computing
- Peer-to-peer

Source: http://en.wikipedia.org/wiki/Cloud_computing
Hardware Virtualization

Benefits: Isolation, Consolidation, and Migration

Virtual Machine Monitor (Hypervisor)

Virtual Machine 1
- User Software
  - App 1
  - App 2
- Windows

Virtual Machine 2
- User Software
  - App 3
  - App 5
- Linux

Virtual Machine 3
- User Software
  - App 4
  - App 6
- Linux

VMWare ESXi
  – Bare-metal hypervisor
  – Advanced virtualization techniques of processor, memory and I/O

Xen
  – Para-virtualization
  – OpenSource
  – Base of some commercial hypervisors

KVM
  – Kernel-based virtual machine is a Linux virtualization subsystem
  – Simpler and smaller hypervisor
<table>
<thead>
<tr>
<th>Service Class</th>
<th>Main Access &amp; Management tool</th>
<th>Service Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>SaaS</td>
<td>Web Browser</td>
<td>Cloud Applications: Social networks, video processing, ...</td>
</tr>
<tr>
<td>PaaS</td>
<td>Cloud Development Environment</td>
<td>Cloud Platform: Programming languages, mashup editors, structured data, scientific applications, ...</td>
</tr>
<tr>
<td>IaaS</td>
<td>Virtual Infrastructure Manager</td>
<td>Cloud Infrastructure: Computer servers, storage, ...</td>
</tr>
</tbody>
</table>

Buyya, Broberg, and Goscinski, *Cloud Computing: Principles and Paradigms*
Deployment Models

Public Cloud
- 3rd party, Multi-tenant, Infrastructure and Services (pay as you go)

Private Cloud
- Runs with the company’s own data center infrastructure for internal/partners use

Hybrid Cloud
- Mixed usage of private and public leasing when private cloud resources are insufficient

Integrate and manage hybrid clouds

Buyya, Broberg, and Goscinski, Cloud Computing: Principles and Paradigms
**Virtual Appliances**

- Application + Environment
  - Operating system
  - Libraries
  - Compilers
  - Databases
  - Applications

This package is a Virtual appliance

Shaped as a VM disk image

- Packaging and Distribution
  - Open Virtualization Format (OVF)
    - File or set of files describing the VM hardware characteristics
  - Virtual Machine Contracts (VMC)

Buza, Broberg, and Goscinski, *Cloud Computing: Principles and Paradigms*
Cloud Essential Features

- **Self-Service**
  - Nearly instant access to resources
- **Per-Usage Metering and Billing**
  - Eliminates up-front commitment
- **Elasticity**
  - Illusion of infinite resources
- **Customization**
  - Multi-tenant clouds have great disparity between environments

Buyya, Broberg, and Goscinski, *Cloud Computing: Principles and Paradigms*
The software toolkit responsible for rapidly and dynamically deploying resources is called the virtual infrastructure manager (VIM).

Features:
- Virtualization Support
- Self-service, On-Demand Resource Provisioning
- Multiple Backend Hypervisors
- Storage Virtualization
- Interface to Public Clouds
- Virtual Networking
- Dynamic Resource Allocation
- Virtual Clusters
- Reservation and Negotiation Mechanism
- High Availability and Data Recovery

Buyya, Broberg, and Goscinski, Cloud Computing: Principles and Paradigms
- Apache VCL
- AppLogic
- Citrix Essentials
- Enomaly ECP
- Eucalyptus
- Nimbus3
- OpenNebula
- OpenPEX
- oVirt
- Platform ISF
- VMWare vSphere and vCloud
Security and privacy affect the entire cloud computing stack
- Third-party services and infrastructure for important data
- Trust in the providers is fundamental
- Local and regulatory issues require attention
<table>
<thead>
<tr>
<th>Problem</th>
<th>HPC</th>
<th>HPC in Clouds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>Buy-and-maintain paradigm</td>
<td>Pay-per-use paradigm</td>
</tr>
<tr>
<td>Performance Optimization</td>
<td>At system acquisition time performance</td>
<td>Joint tuning of applications and system, application-</td>
</tr>
<tr>
<td>System size</td>
<td>under sys. Admin. control</td>
<td>centric performance under user control</td>
</tr>
</tbody>
</table>

R. Aversa, et al., Cloud Computing: Principles and Paradigms
Virtualization

**Industrial strength virtualization coupled with automated resource balancing and virtual image management**

**Entry**

*Basic cloud functions including simple self service interface and infrastructure along with administration & automated deployment*

**Advanced**

*Integrated service management platform with automated IT service deployment, full lifecycle management, metering & chargeback*

**IBM SmartCloud Foundation Infrastructure**

**IBM SmartCloud Entry** delivered by **IBM Starter Kit for Cloud**

**IBM SmartCloud Provisioning** delivered by **IBM Service Agility Accelerator for Cloud**

**TSAM/ISDM/CloudBurst**

**Base Virtualization and Management**
**Image/hypervisor support**

**SKC w/ VMControl 2.3.1.2+**

- **Power systems**
- Provisioning of new servers via SCS (AIX or PowerLinux) or NIM (AIX)
- Standard or Enterprise VMC supported, though Enterprise w/ pooling is recommended for placement across multiple systems
- IP addresses from pool
- Virtual Server properties are defined in OVF from VMControl
- Ability to set CPU/Memory at deploy and change later; ability to add disks post deploy
- Supports multiple VNICS per guest

**SKC w/ VMware 4.1**

- **x86 systems, ESX/ESXi hypervisors**
- Linux or Windows guests
  - Windows guests can have license entered during deploy or on first startup
- Standard or Enterprise VMWare, though Enterprise w/ cluster or resource pool for placement is recommend
- IP addresses from a pool
- Ability to set cpu/memory/disk size at deploy and change later (disk can grow)
- Supports multiple VNICS per guest
Appliances are used as templates for building Virtual Servers or Workloads

- View Virtual Appliance Properties
- Edit Virtual Appliance Properties
- Deploy Virtual Appliance
  - **Basic**, minimal configuration options, including name, description, project, processor information, and memory are displayed
  - **Advanced**, all of the configuration options available for the virtual appliance are displayed.
A workload is a Virtual Server or a deployed Appliance

- Start/Stop
- Capture/snapshot
- Delete
- View workload definition
- Hide
- Copy workload definition
- Resize workload
A Project allows for the grouping of Workloads and Appliances for controlling access.

**Owner** - A project owner has administrator authority to the project and its contents.

**User** - A project user has the authority to use the project and the objects within the project. For example, a project user can deploy a virtual appliance to the project as well as do some limited management of the project and its contents.

**Viewer** - A project viewer has authority only to view the project and the virtual appliances and workloads contained in the project.
Accounts can be created by admins and users can be associated with an account

- Accounts are charged with funds and debited as used
- Charges determined by admin per resource allocated
- Delinquent accounts can be handled via policy (destroy resources, shutdown or do nothing by notify)

Metering records of allocated resources, state and sizes and who allocated them

- Can be used by external/3rd party reporting tools
- Working to make this directly consumable w/ ITUAM

Approvals can be turned on for new deploys and/or resize operations

- If Enabled, users operation is put in as a request
- Admins get emails and can look at open requests
- Notes can be added to the requests by approver or requestor
- Approver can approve or reject request and can modify parms during approval
- Requestor can withdraw or resubmit the request
**Design/Architecture**

**Logical View**

**Biomedical Informatics & Computational Biology**

**Starter Kit Management SW**

- **Starter KIT Self Service UI**
- **REST API**
- **Cloud Foundation Stack Core**
- **CFS On Ramp Adapter Layer**

**Virtualization Scenarios**

- **VMC API Driver**
- **VCenter API Driver**

**Upgrade/migrate Paths for 2012**

- **TSAM)**
- **Orchestration (TSAM-TPM)**

- **Tivoli Provisioning Manager for Images 7.1.1 (P2V-V2V Tools)**

- **Power and x86 – HW Reference Configs**

- **IBM System Director 6.2.x**

- **VCenter 4.1**

- **ESXi**

- **VMware vCenter 4.1**

- **VMControl 2.3.1 (Power)**

- **Power VM**

- **Storage Manager 4.2.1 (Optional for Ref Config)**

- **STG Sys SW**

- **Tivoli Upgrade & Optional Extensions**

- **STG**

- **Tivoli**
**Overall Stack Management**

- Subscriber Level - End User Self Service Interface
- Cloud Admin User Interface
- Isolation of resources and users through projects/workspace
- Approval Policies for workload deployment & resize
- Basic Billing, metering and audit
- Supports underlying virtualization management features
  - resource pooling, workload resiliency
- Support enterprise features of virtual system management
- Storage pools, system pools, network config, placement optimization
- Support Systems Director and VMC for platform management
- Northbound REST APIs (CFS programmatic for partners, services)
- Easy installation of management stack
- Integrate Customer LDAP Server
- VM Power on/off
- Supported on AIX(Power) and on Linux (x86)
- Add v-disk (VMControl)
- Resize disk at deploy (VCenter)
- Supports multiple VM Ware Clusters & resource pools
- Support multiple VMWare data stores
- Support appliances with multiple V-nics
- Admin initiated migration
- Provision via VCenter
- Provision from VS Templates
- Provision to VCenter hosts, & clusters
- Allow/tolerate vSphere initiated migration
- Retrieve deployment properties
- Back-up & Restores via Self Service (VCenter)
- Provides capability to capture Physical to Virtual images for deployment in Starter Kit environment (via TPMfI)
- Provides capability to capture Virtual images and migrate to VMware Images deployable to starter kit environment (via TPMfI)
- Better manage image sprawl - Move the workload from one hypervisor type to a different hypervisor type from a single image source. (via TPMfI)

---

**End User Function**

- Request access to a project
- Request CFS access from login panel
- Request workload creation – appliance deploy
- Set user instance parameters (CPU, Memory)
- Resize Running Workloads
- Delete a Workflow
- Clone a workload
- Start/Stop a workload
- Review workload properties
- Add additional disk (VMControl)
- Set target disk size at deploy (VMware)

**Administrator Functions**

- Configure workload to host
- Configure workload targets to system pool
- Configure Virtual Appliance Parameters
- Register Virtual Appliance for User Selection
- Approve/Reject New Workload Requests
- Approve/Reject Workload Resize Requests
- Configure Billing
- Charging accounts, account assignment
- Configure approvals
- Configure to generate metering records
- Create & Manage Projects
- Add users to projects as admin, deployer, viewer
- Create Users
- Configure Network Pools
- Configure LDAP environment
- Cloud Configuration to VMControl (Power) or VMWare
- Manual Intervention
- Review event logs & failures
- Initiate Workload movement (between projects or via Virtualization manager)
## Platform Support

<table>
<thead>
<tr>
<th>Management node</th>
<th>Managed nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IBM SmartCloud Entry</strong>&lt;br&gt;delivered by IBM Starter Kit for Cloud - Power</td>
<td>Power systems</td>
</tr>
<tr>
<td><strong>IBM SmartCloud Entry</strong>&lt;br&gt;delivered by IBM Starter Kit for Cloud - x</td>
<td>System x (BladeCenter or rack mount)</td>
</tr>
<tr>
<td><strong>IBM SmartCloud Provisioning</strong>&lt;br&gt;delivered by IBM Service Agility Accelerator</td>
<td>x86 servers / RHEL</td>
</tr>
<tr>
<td><strong>TSAM</strong></td>
<td>System x (BladeCenter or rack mount)/ Power systems / System z / other vendors</td>
</tr>
<tr>
<td><strong>IBM Service Delivery Manager / IBM CloudBurst</strong></td>
<td>System x (BladeCenter, or rack mount)/ Power Systems / other vendors</td>
</tr>
</tbody>
</table>
In the 1990's technology started to reach physical limits.

Single powerful processors gave way to commodity processors and a network working as a single machine

Many advances in software development

1990's the decade of the massively parallel supercomputers

Powerful workstations such as the IBM RS 6000

High-Performance computing tackles Grand Challenges

- Scientific & Engineering applications

Technologies developed for High-Performance Computing trickle down and become everyday technologies

GPU went from graphics engines to general purpose computing

Multiple technologies merge and give origin to computing as an appliance

The birth of Cloud Computing