Emerging Technologies and Ideas

Presented by: Prof Mark Baker

ACET, University of Reading
Tel: +44 118 378 8615
E-mail: Mark.Baker@computer.org
Web: http://acet.rdg.ac.uk/~mab

Overview

- Emerging Technologies:
  - Multi-core systems,
  - Cloud-based systems,
  - Virtualisation,
  - Green computing.
- Embracing failure.

Aims and Objectives

- We wish explore the gap between the emerging technologies and innovations as well as the way these can possibly be utilised by the HPC community.
- In particular we want to discuss and debate the codes that are used, and the emerging technologies and innovations…
- Other issues will emerge from the discussion of new emerging technologies that may be relevant to the the HPC community such as robustness and reliability too.

Multi-core Systems

- The number, scale and type multi-core systems available are changing the landscape of application development.
- Parallel computing used to be the preserve of a relatively small group of highly skilled programmers:
  - The emergence of multi-core systems means that parallel programming is becoming a mainstream activity again!
- It is important to ensure that multi-core machines can be effectively and efficiently programmed.
- Also, there is an issue of how to program large clusters of multi-core processes where there is the need for intra and inter communications.
- A key part of that strategy is the availability of high-level tools and libraries to assist expert and the novice parallel programmers create successful programs.

Some Issues

- Can expect 10s-100s of core on processors in the coming years…
- Currently vendors seem to believe that multi-threaded programs will be OK for programming these systems:
  - Vendors think a shared memory programming model is fine!
  - For some programs this may be OK, but coming from a HPC/ parallel computing arena there are issues that threads do not address.
- Architecture of multi-core processors are different – use different cache levels to share data:
  - Consequently will need different strategies for using different processor types.
- Need to create more efficient applications – rather than 20% efficiency, want 90%+ (part of the green computing revolution).

Thoughts

- On large systems you will want to ensure data locality – so you need to have thread affinity.
- Want to effectively load-balance applications running on these systems.
- Need to go back and seriously think about concurrent programming – course and fine grain approaches!
- May want to synchronise threads, (all or groups).
- Will want to optimise strategies for using cache, this may depend on 2^{nd} or 3^{rd} level cache!
- Want to overlap communication and computation too.
- Need to consider the limitations of the architecture when considering communications and access to the system bus.

Thoughts

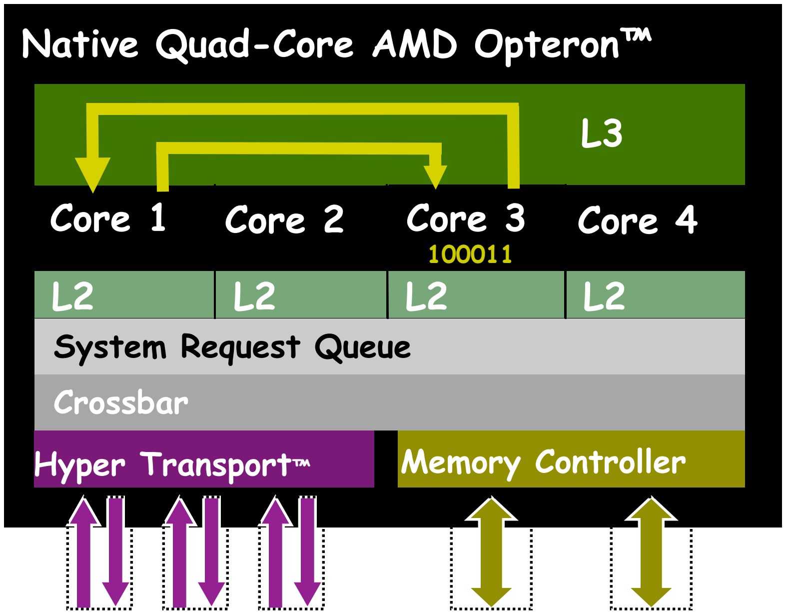
- May need to work with different programming paradigms: message passing, e.g. MPI, OpenMP, through to data – parallelism, e.g. UPC and Fortran, which best suit the application's needs!!
- In the future will need to consider heterogeneous multi-core processors.
- Think about mixed precision programming – single and double precision.
- Lighter-weight threads…
- Need to schedule program efficiently on large machines… do not want fragmented partitioning…

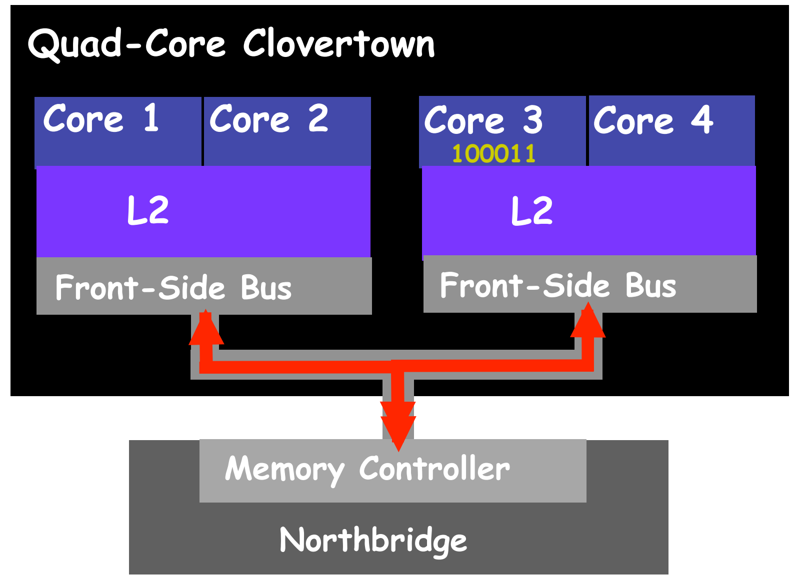
mark.baker@computer.org

Multi-Core: Some Offerings

- AMD
  - Opteron
  - Athlon 64
  - Turion 64
  - Barcelona
- ARM
  - MPCore (ARM9 and ARM11)
- Broadcom
  - SiByte
- Cradle Technologies
  - DSP processor
- Cavium Networks
  - Octeon (16 MIPS cores)
- IBM
  - Cell (Sony, and Toshiba)
  - POWER4,5,6
- Intel
  - Core
  - Xeon
  - Itanium 2
- Microsoft's
  - Xbox 360 (IBM)
- Motorola
  - Freescale dual-core PowerPC
- Picochip
  - DSP devices (300 16-bit processor MIMD cores on one die)
- Parallax - Propeller (eight 32 bit cores)
- HP - PA-RISC
- Raza Microelectronics - XLR (eight MIPS cores)
- Stream Processors
  - Storm-1 (2 MIPS CPUs and DSP)
- Sun Microsystems
  - UltraSPARC IV,
- IntellaSys - seaForth-24.

AMD and Intel Multi-core Processors





AMD: Independent L2 caches:

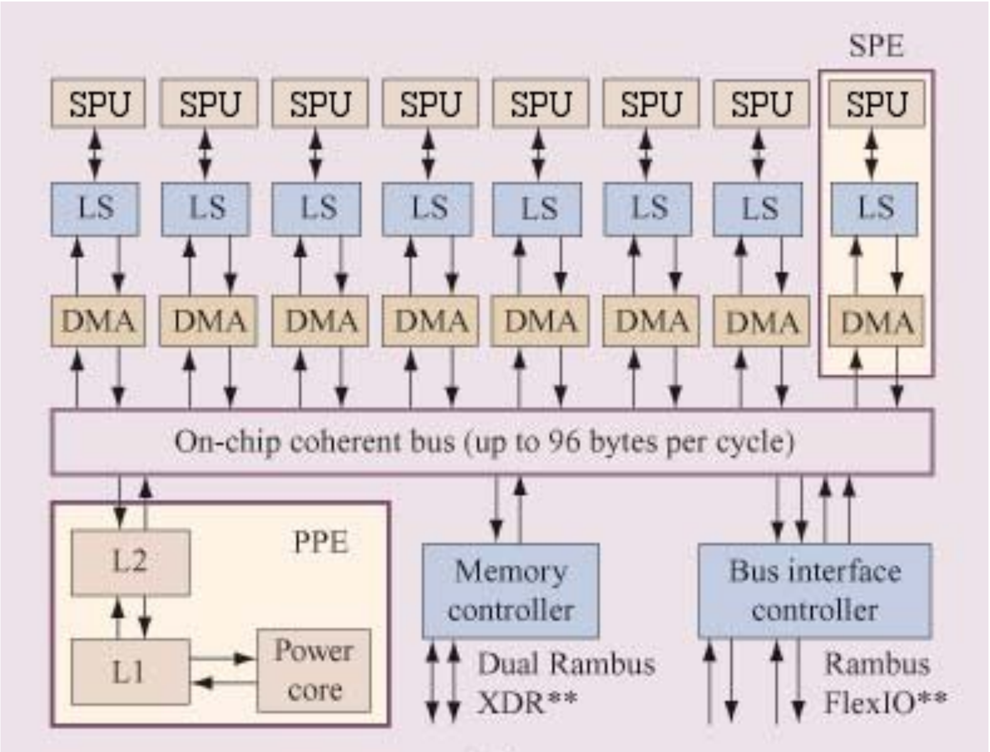
- Multiple Memory modules,
- Communication over Point-to-Point HyperTransport Channels.

Intel: Shared L2 caches:

- Single Memory,
- Communication over Front-side Buses.

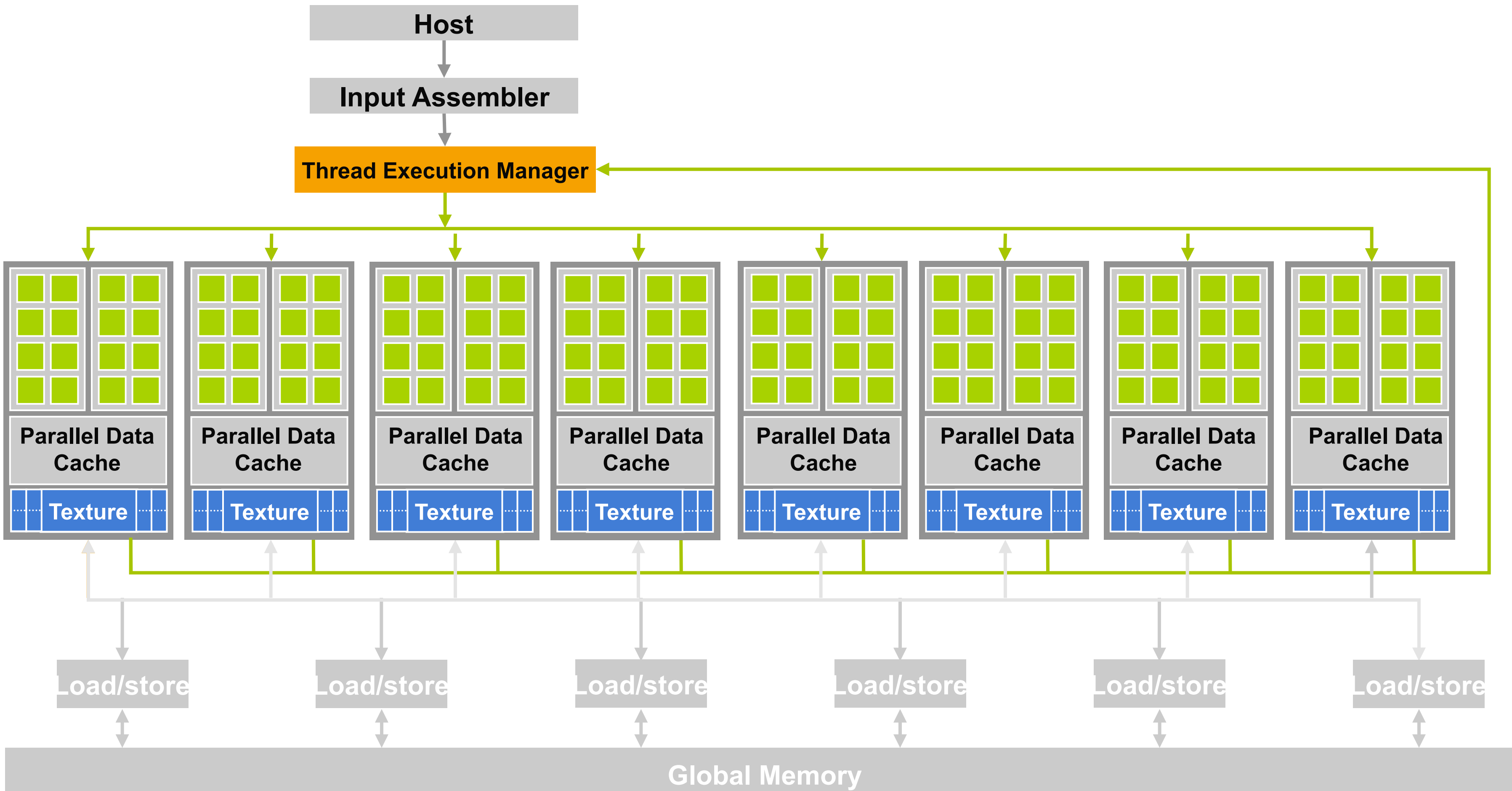
Cell/B.E. Architecture

- Power Processing Element (PPE)
- Eight Processing Elements (SPE):
  - 4 SIMD ALUs
  - DMA Engines
  - 256 kB Local Storage (LS)
- System Memory
  - 25 GB/s
- Element Interconnect Bus (EIB)
  - Over 200 GB/s



GPUs - GeForce 8800

16 threaded SM's, >128 FPU's, 367 GFLOPS, 768 MB DRAM, 86.4 GB/S Mem BW, 4GB/S BW to CPU



Multi-core Issues to Solve

- Communications, synchronsation, resource management between/among cores.
- Debugging: connectivity and synchronisation.
- Distributed power management.
- Concurrent programming.
- OS virtualisation.
- Modelling and simulation.
- Load balancing.
- Algorithm partitioning.
- Performance analysis.

mark.baker@computer.org

Reason for the crisis…

- Memory bandwidth is a major limitation with multi-core systems:
  - All cores must share access to the same memory and therefore have to take turns to read from or write it.
- Disk speed can throttle performance:
  - If a disk can read only 40 MBytes/sec, then a program that needs to read a 400 MByte file will take a minimum of 10 seconds to open it, even if it has no computation to do. Software and scalability
- Software today is frequently not written to take advantage of multiple cores efficiently:
  - It is technically challenging to write efficient code that is free of bugs – a highly scalable program is difficult to build and requires a lot of expertise.

There is MPI style messaging and ..

- OpenMP annotation or Automatic Parallelism of existing software is practical way to use those cores with existing code:
  - As parallelism is typically not expressed precisely, one needs skill to get good performance
  - Writing in Fortran, C, C#, Java … throws away information about parallelism
- HPCS Languages should be able to properly express parallelism but we do not know how efficient and reliable compilers will be:
  - High Performance Fortran failed as language expressed a subset of parallelism and compilers did not give predictable performance.

There is MPI style messaging and ..

- PGAS (Partitioned Global Address Space) like UPC, Co-array Fortran, Titanium, HPJava:
  - One decomposes application into parts and writes the code for each component but use some form of global index.
  - Compiler generates synchronisation and messaging.
  - PGAS approach should work but has never been widely used – presumably because compilers not mature.

What are Clouds?

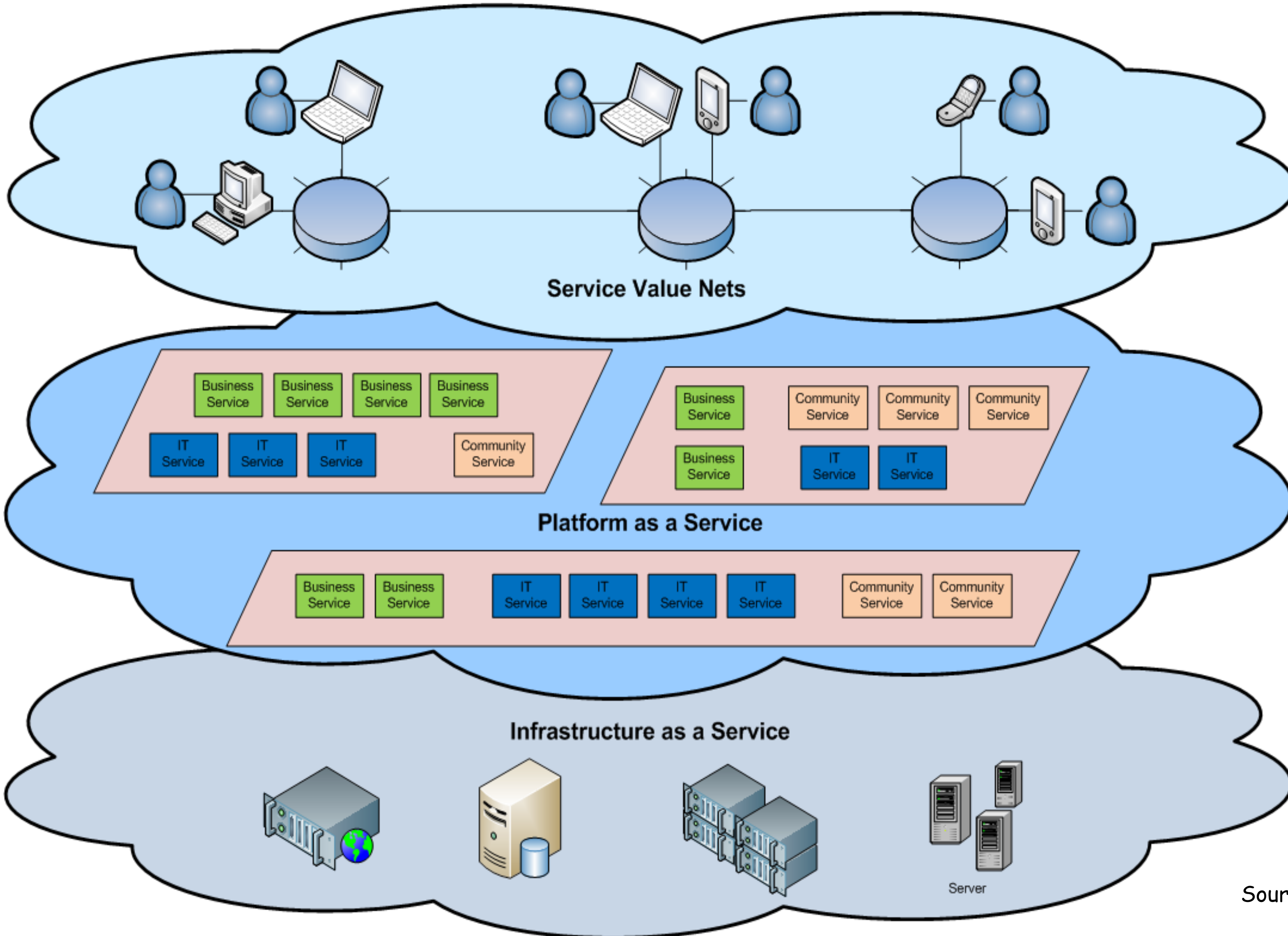
- Clouds are "Virtual Clusters":
  - They may cross administrative domains or may "just be a single cluster"; the user cannot and does not want to know!
- Clouds support access (lease of) computer instances:
  - Instances accept data and job descriptions (code) and return results that are data and status flags.
- When does Cloud concept work:
  - Parameter searches, LHC style data analysis, Face Book …

What makes a Cloud?

- Virtual Machines. ← ***Key Parts of Cloud Definition***
- VM Manager:
  - Scalability.
- File system Infrastructure.
- Remote access (portal).
- Cost?
  - Fairly low cost for CPU/data movement.
- Security?

mark.baker@computer.org

Cloud Architecture



Source: S.Tai

Cloud Computing Service Layers

	Services	Description
Application Focused	Services	Services – Complete business services such as PayPal, OpenID, OAuth, Google Maps, Alexa
	Application	Application – Cloud based software that eliminates the need for local installation such as Google Apps, Microsoft Online
	Development	Development – Software development platforms used to build custom cloud based applications (PAAS & SAAS) such as SalesForce
Infrastructure Focused	Platform	Platform – Cloud based platforms, typically provided using virtualization, such as Amazon ECC, Sun Grid
	Storage	Storage – Data storage or cloud based NAS such as CTERA, iDisk, CloudNAS
	Hosting	Hosting – Physical data centers such as those run by IBM, HP, NaviSite, etc.

Technical Issues about Clouds

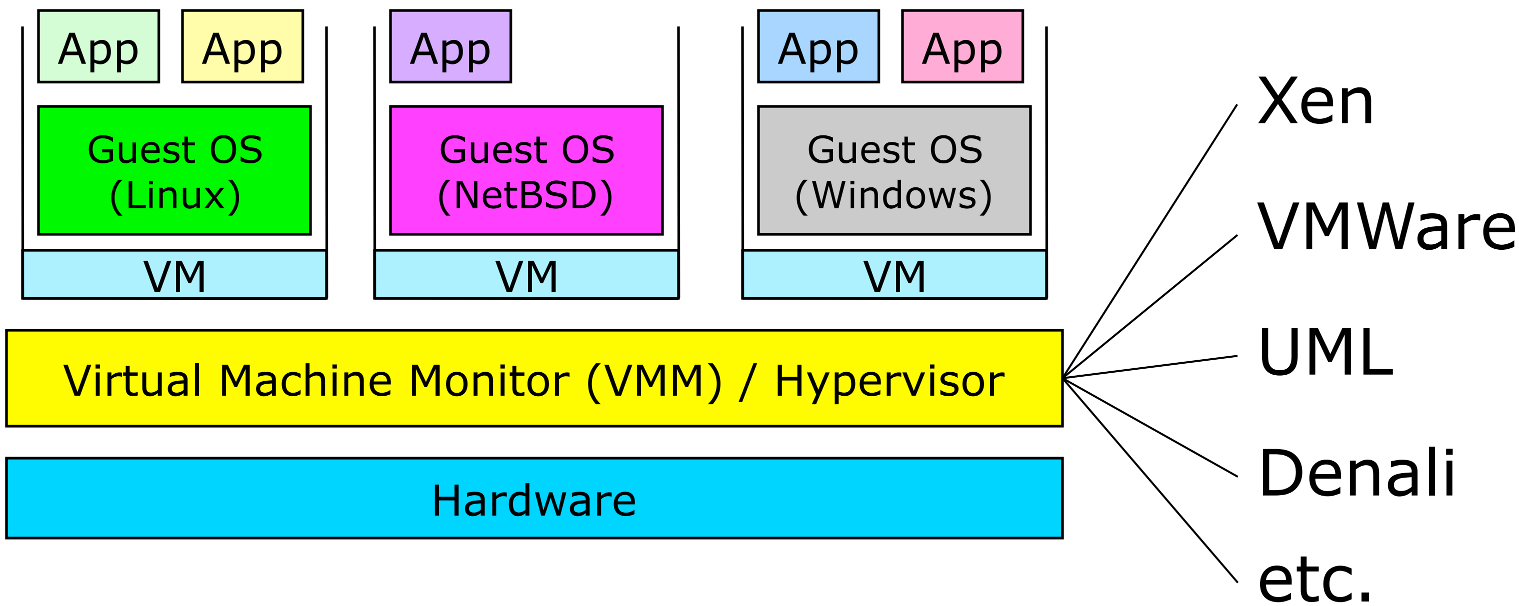
- Using VMs.
- No common standards… APIs often different on each system.
- Protocols – Web Services and RESTful.
- Proprietary workflow systems.
- Database systems.
- Not clear if MPI/OpenMP can be run!
- Multiple VMs on a processor.
- Code development, debugging and optimisation!
- Security.
- Pricing/costs…

Virtualization in General

- Virtual machines run in software that emulates computer hardware:
  - Host machine – hardware running the virtual machine software,
  - Host operating system – operating system running the virtual machine software,
  - Hypervisor – slimmed down host operating system that virtualises the physical hardware,
  - Guest system – operating system.
- Examples of Virtual Machines:
  - VMware,
  - Microsoft Virtual PC and Microsoft Virtual Server,
  - Parallels Workstation,
  - Sun xVM,
  - Kernel-based Virtual Machine (KVM),
  - Xen (Opensource),
  - ++++

Virtual Machines

- VM technology allows multiple virtual machines to run on a single physical machine.



Performance: Para-virtualization (e.g. Xen) is very close to raw physical performance

Virtualization in General

- Advantages of virtual machines
  - Run operating systems where the physical hardware is unavailable
  - Easier to create new machines, backup machines, etc.
  - Software testing using "clean" installs of operating systems and software
  - Emulate more machines than are physically available
  - Timeshare lightly loaded systems on one host
  - Debug problems (suspend and resume the problem machine)
  - Easy migration of virtual machines (shutdown needed or not)

Virtual Machines

- Multiple virtual machine instances on a single physical host:
  - Fault tolerance, Isolated OS instances, and Virtual servers.
- Clouds are using Para-virtualization – which has Virtual Machine Manager and an Hypervisor.
- Many HPC application are only 15 -20% efficient on bare-metal… many VM manufacturers feel that using VM are more efficient – up to 70 – 80%.
  - We are running a range of benchmarks (HPC and others) on bare-metal and VM, to see if using VMs are more efficient!
- In addition, using VM means that you can control the amount of energy used by the applications running – need to create various tools and utilities to manage the systems.

Workload Consolidation pros/cons

- Pros:
  - Each application can run in a separate environment delivering true isolation,
  - Cost Savings: Power, space, cooling, hardware, software and management,
  - Ability to run legacy applications in legacy OSs,
  - Ability to run through emulation legacy applications in legacy HW.
- Cons:
  - Disk and memory footprint increase due to multiples OSs,
  - Performance penalty caused by resource sharing management.

- Workload consolidation provides the basis most usages/benefits of virtualization

Green Computing

- Why?
  - Computer energy is often wasteful:
    - Leaving the computer on when not in use (CPU and fan consume power, screen savers consume power).
  - Pollution:
    - Manufacturing techniques,
    - Packaging,
    - Disposal of computers and components.
  - Toxicity:
    - Toxic chemicals used in the manufacturing of computers and components which can enter the food chain and water!

Embracing Failure

- As more systems encompass ever-increasing numbers of components, even a small fault rate on individual processors will generate multiple faults across the components, stopping long-running applications in their tracks.
- Weather/Climate Modelling:
  - Outlive Complexity:
    - Increasingly sophisticated models,
    - Model coupling,
    - Inter-disciplinary.
  - Sustained Performance:
    - Increasingly complex algorithms,
    - Increasingly diverse architectures,
    - Increasingly demanding applications.

mark.baker@computer.org

Key Challenges

- How do you know if there were any transient or permanent failures of the hardware or system software that invalidate the computation? (Detect)
- Can a middleware library implement the functionality of the computational workflow model – isolating the computational software from the transient and permanent failure management? (Isolate/Contain)
- In the case of permanent failures, how does one remap the computation to the remaining available resources? Does the application programmer have to do this? … (Recover)

mark.baker@computer.org

Summary

- Multi-core systems – need to understand underlying hardware, and be able to efficiently program clusters of multi-core processors.
- Cloud-based systems – becoming increasing popular – Grid is dead, and clouds are becoming popular!
  - Flexible utility-based platform, that has a range of interesting and innovative properties.
- Virtualisation – being using in Clouds, but also to provide management and control of machines.
- Green computing – becoming important, and a key feature of FP7.
- Embracing failure – machines are becoming bigger, more sophisticated and so failure is increasingly like – must check out ways of creating reliable applications!