CS838 - Virtualization

The Datacenter as a Computer
Class Notes

February 6, 2013

- Chapter 2
  - **Chapter Overview:** Chapter 2 focuses on how the design process for the WSC stack differs from the design process for desktop computers, both at the hardware and software layers. Platform-level software, cluster-level infrastructure, and application-level software are all designed around the unique profile of WSC applications, including by harnessing the inherent data- and request-level parallelism of Internet applications. Cluster-level infrastructure software must be created to manage the WSC's resources, much like an OS manages a single physical machine's resources. Building hardware abstractions, programming frameworks, and a monitoring infrastructure help ease the software development process for WSC-based applications and provide support fault tolerance and workload churn.
  - Many different levels of software - desktop, cluster, datacenter, warehouse scale computer
  - At the platform level: firmware and kernel software
  - At the cluster level: distributed systems software that abstract across data-center resources to provide uniform access to those resources, such as MapReduce, Hadoop, etc.
  - At the application level: software for particular services, such as Gmail or Google Maps
  - Different types and levels of parallelism can be exposed at higher layers of hardware
    - At Datacenter/WSC levels (very easy): *data level parallelism* and *request level parallelism*
      - Data level parallelism due to very large amounts of separate data records to be processed, which can be processed independently, such collections of billions of web pages for Search
      - Request level parallelism due to large number of independent requests to the system from many users accessing the same system at once
      - Remzi: This is important: with a particular algorithm, it's often hard to decompose it into different parallel parts. When you have data or request parallelism, these are very easy kinds of parallelism to deal with. This is why when we have these kinds of parallelism, they enable applications like Search, which are inherently parallel applications.
    - At desktop levels: parallelism is much more difficult to achieve because of the workloads in place at these levels
    - Cluster levels:
      - Resource management
      - abstract lower level data structures
      - maintenance - apply patches and updates to lower level nodes
    - Application levels:
      - Hardware is assumed general purpose
      - If hardware changes software cannot be affected
  - Homogeneous hardware is used, doesn't design HW for specific specifications
  - Assume hardware is faulty, which is OK if homogeneous hardware is utilized
  - Have a toolbox of software:
    - Replication, sharding, load-balancing, health checking and watchdog timers, integrity checking, application-specific compression, eventual consistency

    * Using the toolbox should make using the hardware more effective
    * Interesting note is application-specific compression: this is perhaps the strangest/most interesting, and arises because memory capacity at the WSC level is still limited, but to minimize latency appications need to fit the largest part of the working set possible into memory. The trips from memory to CPU to compress/decompress parts of the application introduce overhead that's much less than the cost of going to disk. But the compression needs to be application-aware and tailored, or else it can introduce unnecessary overhead.
    * Integrity checks: can't check everything, but if you are consistently checking multiple layers issues should be mitigated
  - Interesting notes:
    * update cycle is on a weekly basis, transparent to user
    * kernel used in datacenter is very old versus application layers (2-3 years)
    * application-layer software assumes that hardware on which it runs is general-purpose. Reason: by the time the application is launched, hardware will have to be changed, so application-specific hardware doesn't make sense.

- Chapter 3
  - **Chapter Overview:** Chapter 3 focuses on the choice of hardware building blocks for WSCs, which are characterized by a large and homogeneous hardware base of server hardware, networking fabric, and storage hierarchy components. The paper compares a high-end 128 processor-core SMP server with a mid-range quad-core server, and determines that although the SMP server exhibits greater parallelism and better performance on a workload that can be handled entirely within one SMP server, as soon as you compare clusters of high-end SMP servers with similar clusters of mid-range servers, the latter has a better price/performance rating. Given the scale of WSC workloads, it is therefore more effective to use mid-range servers as building blocks. Smaller low-end servers are also undesirable as they require a larger software-engineering effort to handle the increased need for explicit parallelization (where it would have been handled implicitly by multi-core mid-range servers), and lead to increased networking costs and lower per-server utilization.
  - trade-offs between what kinds of individual nodes to use in the WSC
  - Enterprise level nodes provide greater possible parallelism but at a greater cost. Lower-end nodes provide better cost efficiency, but don't handle parallelism on large data as well.
  - If you want more parallelism, the enterprise server has great parallel performance. BUT, at the scale they're working at, the parallelism is so large and demands so many nodes, that the improvement of having these large enterprise server isn't much.
  - Argument: the price premium you pay to get the enterprise server isn't justified by the performance improvement you get for most applications, so it's better to use lower-end servers.
  - On the other hand, if the low-end server is preferable to enterprise servers, why not take it to the extreme and use super low-end machines that are cheaper. BUT, this isn't good either, because you then need to devote a lot of software-development resources to address parallelism needs, reduce latency from increased networking among machines, etc. The greater number of networked small machines also adds concurrency/synchronization overhead.
  - Re: the Dick Sites talk that happened the previous day: at Google they embrace the philosophy of buying lots of cheap commodity parts and addressing the drawbacks via internal engineering solutions. Is this good? Depends on your perspective.

- Chapter 4
  - **Chapter Overview:** Chapter 4 focuses on WSC power and cooling requirements, which are substatial, and on the need to strike a balance between construction costs and reliability when building a WSC. Most WSCs have redundant power sources mediated by a UPS (uninterruptable

power supply), and are built near a utility's substation for efficient access to the energy source. They also have redundant cooling systems and paths, ranging from older techniques such as CRAC units which cool air through an element, to newer techniques such as free cooling, in-rack cooling, and building container-based datacenters.
  - Power and cooling - datacenter is basically a unit which takes in power and outputs heat
  - Power:
    * Main source of power directly from utility typically at medium voltage
    * Backup power provided on-site by generators
    * Uninterruptable Power Supply (UPS) provides power to server racks,
    * Battery backups bridge gap between utility power and generators during power failure
  - Cooling:
    * CRAC (Computer Room Air Conditioning) typically used (raised floors push air to server racks, hot and cold aisles output heat from building). You should alternate hot and cold aisles, and other strategies. This is a relatively old technology/approach.
    * Free Cooling - use environment to help cool servers - instead of actively cooling hot air by pushing it through an element, just circulate it outside for awhile to cool it down. This only works in certain environments. Works great until it doesn't work then you need CRAC anyway.

- Chapter 5
  - **Chapter Overview:** Chapter 5 focuses on how to interpret power consumption versus computational work performed by a WSC which provides a notion of the energy efficiency. They talk about losses in energy efficiency from the standpoint of infrastructure enhancements to the datacenter. However, many enhancements must be tested against known benchmarks which are difficult to deploy in used datacenter systems. They also focus on the power system of the datacenter and how distribution efforts lead to energy efficiency loss.
  - Efficiency of WSC defined as $\frac{valuable\ output}{total\ power}$ typically PUE is metric used
  - Overly large source of power consumption is cooling, study found that increasing temp by 10 degrees would be OK
  - They found that most of their servers run at 10-50% of their max load, which is a mismatch with the efficiency of servers, which run most efficiently at peak load. Due to load balancing and their distributed algorithms though, the servers rarely operate at peak load.
  - So, they need to add energy proportionality as a design goal for their systems. Disk is unlikely to be energy-proportional, because it's always spinning. You could turn it off, or spin it slower during periods of low use. In general, it's desireable to put equipment in idle or low-power mode when demand is low. But this takes a performance hit when you need to fire it up again.
  - *Energy Proportionality* - scaling relationship between power consumption and utilization
    * leads to issue because turning off things doesn't work well (disk is big issue here - what to do? turn it off, spin down disks)
  - Can't turn off services altogether
  - Turning off servers leads to needing to turn on more to make up for usage spikes to minimize latency
  - Replication adds to issues since energy goal is to minimize number of nodes that are on, but replication goal is to keep as many things running as possible

- Chapter 7
  - **Chapter Overview:** Chapter 7 focuses on failure and fault tolerance issues of a datacenter. Talk about hardware failure cases including power supplies, fans, HDD failures, as well as "soft" errors which are mitigated by ECC memory. A focus is on the type of failure which has occurred and how the system can be constructed to keep fault tolerance high.

  - Software detects and masks hardware faults
  - Four levels of faults: corrupted, unreachable, degraded, masked
  - Goal should be to degrade gracefully which allows access to the server for as long as possible
  - Also allows putting off repairs to do in batches instead of individually in response to failure
  - 4 hour required repair 2-3x more expensive then 24 hour repair
  - Many different types of faults
    * DRAM soft-errors
    * disk errors
    * leads to shorter restart times. 1% of faults take more than a day to repair and get the system back up and running.
  - Number of restart times should be low, interesting to note that once hardware fails it is more likely to fail but usually restarts solve issue
  - Modeling to predict failures is useless
  - **Gray Study** [35] - Tandem failure leads to the conclusion: it's not the hardware that's at fault, it's the software.
  - Usually software, configuration, people at fault - not so much the hardware
  - Fault tolerance at software layer helps manage servers, taking down one node for repair is OK for overall system.
  - Increasing the software base could lead to more virtualization

- Chapter 1/8
  - **Chapter Overview:** Introduction and conclusion of the paper. Intro forms the basis for what a warehouse scale computer is versus a datacenter as well as answering the how, why, where questions. The conclusion wraps up facts from the paper as well as future challenges of implementing WSCs.
  - Defines warehouse scale computer and how it differs from a datacenter
  - Datacenters are used by many different companies where as WSC is used by one entity
  - WSC should be viewed as a single computer from a high level. WSCs are built from relatively homogeneous components and use common software and scheduling tools across all nodes to handle many workloads.
  - WSC should consist mostly of commodity hardware and should feature a highly fault-tolerant software layer so that hardware failures aren't an issue. Still, certain subsystems such as SQL databases might be better to have on high-end SMP servers.
  - Software for WSC differs greatly both from traditional client/server model, and from high-performance computing models, in a number of ways: parallelism, workload churn, platform homogeneity, fault-free operation, usual medium utilization (as opposed to HPC).
  - Energy proportionality should be a primary design goal for WSCs.
  - Challenges: rapidly changing workloads, building balanced systems from imbalanced components, energy efficiency, challenges presented by Amdahl's law.
  - Amdahl's law allows us to focus on increasing thread and core count but this will lead to issues:
    * Focus on improving components other than CPUs as they will become the bottlenecks (DRAM, disks, networking fabric).
    * Overall bottleneck will become networking layer for parallelized tasks.