Eras of Operating Systems

<table>
<thead>
<tr>
<th>Year</th>
<th>Era</th>
</tr>
</thead>
<tbody>
<tr>
<td>1950</td>
<td>Batch</td>
</tr>
<tr>
<td>1960</td>
<td>Interactive</td>
</tr>
<tr>
<td>1970</td>
<td></td>
</tr>
<tr>
<td>1980</td>
<td>(L) Distributed</td>
</tr>
<tr>
<td>1990</td>
<td>(L)</td>
</tr>
<tr>
<td>2000</td>
<td>Cloud-Mobile</td>
</tr>
<tr>
<td>2010</td>
<td></td>
</tr>
</tbody>
</table>
Eras of Operating Systems

<table>
<thead>
<tr>
<th>Year</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1950</td>
<td>Batch</td>
</tr>
<tr>
<td>1960</td>
<td>Interactive</td>
</tr>
<tr>
<td>1970</td>
<td>(L) Distributed</td>
</tr>
<tr>
<td>1980</td>
<td>(I) Distributed</td>
</tr>
<tr>
<td>1990</td>
<td>Cloud-Mobile</td>
</tr>
<tr>
<td>2000</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
</tr>
</tbody>
</table>
### Eras of Operating Systems

<table>
<thead>
<tr>
<th>Year</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1950</td>
<td>Batch</td>
</tr>
<tr>
<td>1960</td>
<td>Interactive</td>
</tr>
<tr>
<td>1970</td>
<td>(L) Distributed</td>
</tr>
<tr>
<td>1980</td>
<td>(I) Distributed</td>
</tr>
<tr>
<td>1990</td>
<td>Cloud-Mobile</td>
</tr>
<tr>
<td>2000</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
</tr>
</tbody>
</table>

“computer utility”
Eras of Operating Systems

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch</td>
<td>Interactive</td>
<td>(L)</td>
<td>Distributed</td>
<td>(I)</td>
<td>Cloud-Mobile</td>
<td></td>
</tr>
</tbody>
</table>

(L) Distributed (I)
Eras of Operating Systems

<table>
<thead>
<tr>
<th>Year</th>
<th>Era</th>
</tr>
</thead>
<tbody>
<tr>
<td>1950</td>
<td>Batch</td>
</tr>
<tr>
<td>1960</td>
<td>Interactive</td>
</tr>
<tr>
<td>1970</td>
<td>(L)</td>
</tr>
<tr>
<td>1980</td>
<td>Distributed</td>
</tr>
<tr>
<td>1990</td>
<td>(I)</td>
</tr>
<tr>
<td>2000</td>
<td>Cloud-Mobile</td>
</tr>
<tr>
<td>2010</td>
<td></td>
</tr>
</tbody>
</table>

- **Batch**
- **Interactive**
- **Distributed**
- **Cloud-Mobile**
- **Personal Computers**
Eras of Operating Systems

1950 - Batch
One job at a time

1960 - Interactive
Many jobs sharing

1970 - (L)
Personalized immersive world managing work (desktop)

1980 - Distributed

1990 - (I)

2000 - Cloud-Mobile
Personalized immersive world managing life and social relations

2010
Eras of Operating Systems

1950: Batch
1960: Interactive
1970: (L) Distributed
1980: (I)
1990: Cloud-Mobile
2000: Internet
2010:

“OS interfaces with”:
- TELNET
- FTP
- SMTP
- RLOGIN
- RCOPY

“OS integrates with”:
- Protocol software
- IPC, RPC
- Daemon processes
- Client-server, X-windows
- Hyperlink, URL
- Browser
- Search
Eras of Operating Systems

- 1950: Batch
- 1960: Interactive
- 1970: (L)
- 1980: Distributed
- 1990: (I)
- 2000: Cloud-Mobile
- 2010: Protection-security, languages, abstraction, memory management, files, fault tolerance, virtualization, parallel computing, network, cloud
Eras of Operating Systems


Batch  Interactive  (L)  Distributed  (I)  Cloud-Mobile

Protection-security, languages, abstraction, memory management, files, fault tolerance, virtualization, parallel computing, network, cloud

OS principles in education
Eras of Operating Systems

- **1950**: Batch
- **1960**: Interactive
- **1970**: (L)
- **1980**: Distributed
- **1990**: (I)
- **2000**: Cloud-Mobile
- **2010**: Cloud-Mobile

Protection-security, languages, abstraction, memory management, files, fault tolerance, virtualization, parallel computing, network, cloud

OS principles in education

Capabilities
Power behind Advancement

Pushing

Pulling

Apple

Windows

iOS

Android
Advancement of Hardware

- Many core architecture
- Heterogeneous core architecture
  - GPGPU
  - Big/little
- Wimpy server architecture
- NVMe storage devices
- Storage class memory
- Finer fabrication process
- Faster and smarter bus and interface architecture
Many Core Architecture

- Scalability matters
- Conventional SMP OS architecture induces a significant amount of inter-core interference
- OSs with little inter-core relations
  - ie) Popcorn Linux
Heterogeneous Core Architecture

- GPGPU
  - Job scheduling
  - Preemption
  - Common interface

- Big/little or heterogeneous cores
  - Job distribution
  - Power management
Wimpy Server Architecture

- ARM released server-class line-ups
- Intel and AMD are pushing their embedded chips to server market
- Will they earn their market share?
- Challenges
  - Large scalability
  - Fault-tolerance
  - Data management
- FAWN cluster, published in SOSP ‘09
NVMe Storage Devices

- Storage device is no longer a bottleneck

![Access Latency Chart]

- Approaches
  - User-level I/O stack
  - I/O virtualization
  - Separation of control plane and data plane

Table 1: Server node configuration.

- Processor: Xeon E5-2690, 2.9GHz, dual socket-8 cores
- HDD Storage: 1× 15K SAS Enterprise disk
- SSD Storage: 4× Samsung 843 Pro SATA SSD (2013)
- NVMe Storage: 1× Samsung XS 1715 NVMe (2014)
- Memory Capacity: 64 GB ECC DDR3 R-DIMMs
- Memory Bandwidth: 102.4GB/s (8 channels of DDR3-1600)
- RAID Controller: LSI SAS 2008 (up to 290,000 IOPS)
- Network: 10 Gigabit Ethernet NIC
- Operating system: Ubuntu 12.04.5
- Linux Kernel: 3.14 Mainline
- FIO Version: 2.1.10 run with direct I/O
- HammerDB version: 2.16
- MySQL version: 5.5
- Cassandra version: 2.0.9
- MongoDB version: 2.6.0

The TPC-C workload is organized as warehouses within a company, where each warehouse includes a predefined number of districts and customers. It supports five representative transactions: two are strictly read-only, while three perform both read and write access (Dell 2013). All transactions operate against a database of nine tables; the workload also defines the overall distribution of transactions and their response time requirements (TPC 2010). TPC-C measures the number of new orders processed per minute, and the metric is expressed in transactions-per-minute (tpmC).

4.1.1 TPC-C Experimental Setup and Optimizations

We use HammerDB (HammerDB 2014) to generate the schema and transactions and MySQL (Oracle 2014) as the underlying database.

Our initial setup and experiments indicate sub-optimal performance on stock installation of MySQL, thereby prompting the need to identify and optimize several parameters. In interest of space, we only summarize four key parameters that had the most impact on performance:

1. Concurrent Connection Limit: We set the number of concurrent connections supported in MySQL and the number of open file descriptors in the Linux kernel to 32K.
2. I/O scheduler: We use the `noop` Elevator scheduler (see Section 2) to gain optimal SSD performance.
3. Thread Concurrency: Number of concurrent threads inside MySQL's storage engine (InnoDB) is set to match the maximum supported CPU threads (32).
4. Buffer Pool Size: We use a buffer pool size of 8 GB for caching InnoDB tables and indices.

We initialize the database with 1024 warehouses, resulting in a 95 GB dataset. As mentioned in section 4, we experiment with a single SSD and a four SSD setup. The SSD experiments are subsequently compared with a single NVMe drive, and a RAM-based tmpfs filesystem.

4.1.2 TPC-C Performance Evaluation

We use timed test driver in HammerDB and measure results for up to six hours. To establish a stable TPC-C configuration, we first explore the throughput of TPC-C system by scaling the number of virtual users (concurrent connections), as shown in Figure 6. While maximum throughput is achieved at ten virtual users, increasing concurrency past that point leads to a sharper fall in throughput. We observe more consistent performance between 60-65 virtual users. Based on these sensitivity results, we select 64 virtual users for all experiments.

TPC-C is a disk intensive workload, and is characterized as a random mix of two reads to one write traffic classification (Dell 2013). While it mostly focuses on the overall throughput metric, lower latency storage subsystem reduces the average time per transaction, thus effectively increasing the overall throughput. Figure 7 shows the I/O latency impact on effective CPU utilization for all four previously described experimental categories. As shown in the figure, for the baseline single-SSD configuration, the system spends most of its time in I/O wait state. This limits the throughput of the system as CPU spends the majority of its execution cycles waiting on I/O requests to complete. Figure 8 shows the disk read and write bandwidth for the SSD configuration, with writes sustaining at about 65 MB/s and reads averaging around 140 MB/s. While NAND flash has better latencies than HDDs, write traffic requires large management overhead, leading to a significant increase in access latencies.
Storage Class Memory

- Entire memory area is non-volatile
- Will their time really come?
  - Intel X-point provides 10 times larger memory
- No caching and buffering are required
- No separation between applications and data
  - Why do we need files and file systems?
Finer Fabrication Process

- Improved energy efficiency
  - Diminishing return of DVFS
  - Increased portion of leakage power
  - What will be the power management of future?
- Large cache size for multiple cores
  - Diminishing return of cache size increment
  - What will you use such abundant resource for?
  - What will be the smart ways to fully utilize them?
Bus and Interface Architecture

- Inter-device direct communications
- IO-MMU
- SR-IOV and MR-IOV technologies
- And so on...
Expansion of Services and Applications

- On the large end
  - Federated cloud computing
  - Exa-scale computing
  - Stream data processing and deep learning workloads
  - Terra-swarm, fog computing or edge computing

- On the small end
  - Energy-thermal efficient mobile computing
  - IoT service support
  - UHD and VR support
  - User interface
Conclusion

- Operating systems are continuously changing
- User needs are pulling while technologies are pushing
- What will the future operating systems look like?