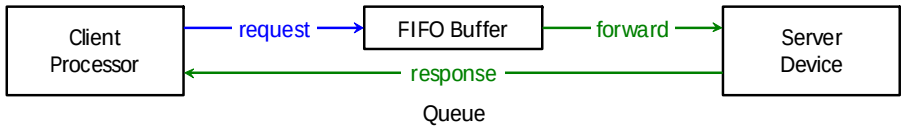
Queuing Theory:
Memory Buffer
Limits on Superscalar Processing

Client/Server Model of I/O



Fast CPU is client for slower I/O services

Buffer stores client requests and forwards at slower server response rate

Latency

Time between client request and buffer server response

Throughput

Number of services provided per unit time

Bandwidth

Maximum data transfer rate of I/O channel (including buffer)

Capacity

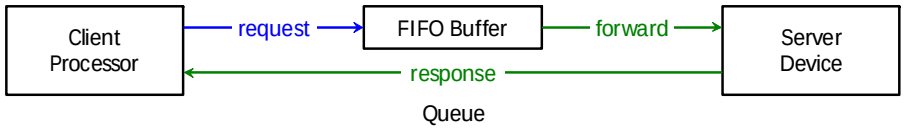
Maximum throughput of server through buffer

Depends on bandwidth and service rate (server speed)

Utilization

Request rate as proportion of capacity

Buffer Operation



Client requests

Intermittent high speed requests (**bursty**)

Peak client request rate >> **Average** client request rate

Service responses

Peak client rate > Server rate > **Average** client rate

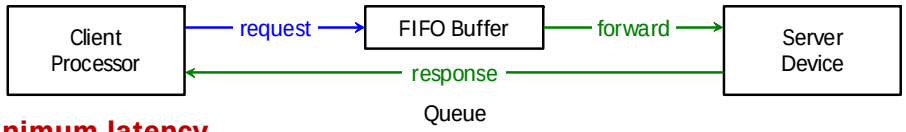
FIFO buffers requests in order of arrival

Stores requests arriving at higher client request rate

Forwards requests to server at lower server response rate

Request forwarding rate = server response rate

Buffer Latency and Overflow



Minimum latency

Determined by maximum service response rate

Buffer overflow

Buffer fill rate = Client Request Rate – Service Response Rate

Buffer fills continuously for too long ⇒ buffer overflow

Example

Peak CPU disk read rate = 1 read/cycle = 10^9 read requests/second

Disk can provide = 10^7 responses per second = 100 CPU cycles/read

CPU sees minimum latency of about 100 CPU cycles

Buffer can hold 1000 requests

Continuous requests ⇒ overflow in $1000/(10^9 - 10^7) \sim 10^{-6}$ seconds

Utilization — Latency Trade-Off

Utilization

Higher client request rate

More services per second ⇒ Higher utilization

Server cannot work faster (service rate is fixed)

More requests are buffered ⇒ longer queue length (higher buffer level)

Total latency for one request = server latency + queuing time

More requests in buffer queue ⇒ longer queuing time

Higher utilization ⇒ Higher total latency

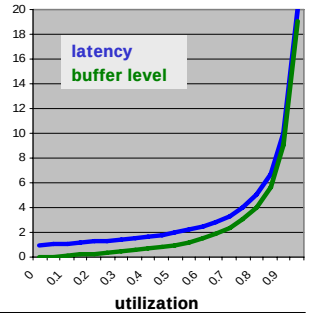
Buffer overflow

Average request rate > average response rate

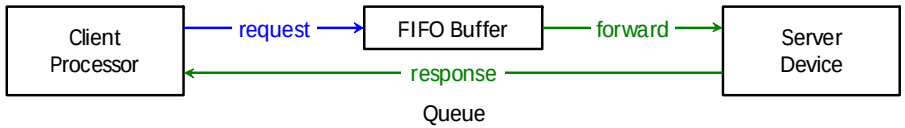
More requests enter buffer than leave

Buffer level rises

After long time, buffer overflows



Queuing Theory



Assumptions

Client Requests

Arrive independently (Poisson statistics)

Have random length (bytes to transfer)

Buffer

Stores requests and forwards in order of arrival (FIFO) at service rate

Reaches an average "steady state" level of stored requests

Server

Provides services to each request independently (Poisson statistics)

Results

Average latency = 1 / (average service rate – average arrival rate)

Average buffer level = average arrival rate × average buffer latency (Little's Law)

Poisson Statistics

Random events

Occur independently

Average total number of events grows linearly with time

$$P_t(k) = \text{probability}(k \text{ events in time } t) = \frac{(\alpha t)^k}{k!} e^{-\alpha t}$$

$$p(t) = \text{probability}(\text{time interval between 2 events is } t) = \alpha e^{-\alpha t}$$

$$\langle t \rangle = \text{Average time between events} = \int_0^\infty t\left[\alpha e^{-\alpha t}\right] dt = \frac{1}{\alpha}$$

α = average rate of events

$$\langle k \rangle = \text{Average number of events} = \sum_{k=0}^{\infty} k\, P_t(k) = \alpha t$$

αt = average event rate × time interval

Request and Response Statistics

λ = average request rate

Probability of k requests in time $t = \frac{(\lambda t)^k}{k!} e^{-\lambda t}$

Distribution of times between requests $= \lambda e^{-\lambda t}$

$\frac{1}{\mu}$ = average volume (bytes) per response

C = server capacity (maximum response volume per second)

$$\mu C = \frac{\text{server volume / second}}{\text{average volume / response}} = \text{average response rate}$$

Probability of k responses in time $t = \frac{(\mu C t)^k}{k!} e^{-\mu C t}$

Distribution of server response times $= \mu C e^{-\mu C t}$

Buffer State

$B(t+\Delta t)$ = requests in buffer at time $t+\Delta t$

Possible cases for $B(t+\Delta t)=n$

1. $B(t)=n-1$ — $1 \rightarrow [\,n-1\,] \rightarrow 0$

   During $[t,t+\Delta t]$ 1 new request and 0 finished responses

2. $B(t)=n+1$ — $0 \rightarrow [\,n+1\,] \rightarrow 1$

   During $[t,t+\Delta t]$ 0 new requests and 1 finished response

3. $B(t)=n$ — $0 \rightarrow [\,n\,] \rightarrow 0$

   During $[t,t+\Delta t]$ 0 new request and 0 finished responses

4. $B(t)=n$ — $1 \rightarrow [\,n\,] \rightarrow 1$

   During $[t,t+\Delta t]$ 1 new request and 1 finished response

Buffer State Transition

$P_n(t+\Delta t)$ = probability$\left(B(t+\Delta t)=n\right)$

$$= \text{probability}\begin{Bmatrix} B(t)=n-1 \textbf{ AND } 1 \text{ request } \textbf{AND } 0 \text{ finish} \\ \textbf{OR } B(t)=n+1 \textbf{ AND } 0 \text{ request } \textbf{AND } 1 \text{ finish} \\ \textbf{OR } B(t)=n \textbf{ AND } 0 \text{ request } \textbf{AND } 0 \text{ finish} \\ \textbf{OR } B(t)=n \textbf{ AND } 1 \text{ request } \textbf{AND } 1 \text{ finish} \end{Bmatrix}$$

$$\begin{aligned} &= P_{n-1}(t)\,P_{\Delta t}(1 \text{ request})\,P_{\Delta t}(0 \text{ finish}) \\ &\quad + P_{n+1}(t)\,P_{\Delta t}(0 \text{ requests})\,P_{\Delta t}(1 \text{ finish}) \\ &\quad + P_{n}(t)\,P_{\Delta t}(0 \text{ requests})\,P_{\Delta t}(0 \text{ finish}) \\ &\quad + P_{n}(t)\,P_{\Delta t}(1 \text{ request})\,P_{\Delta t}(1 \text{ finish}) \end{aligned}$$

Buffer State Transition

$$P(1 \text{ request in } \Delta t) = \frac{(\lambda\Delta t)^1}{1!}e^{-\lambda\Delta t} = \lambda\Delta t + o\left((\Delta t)^2\right)$$

$$P(0 \text{ request in } \Delta t) = \frac{(\lambda\Delta t)^0}{0!}e^{-\lambda\Delta t} = 1-\lambda\Delta t + o\left((\Delta t)^2\right)$$

$$P(1 \text{ finish in } \Delta t) = \frac{(\mu C\Delta t)^1}{1!}e^{-\mu C\Delta t} = \mu C\Delta t + o\left((\Delta t)^2\right)$$

$$P(0 \text{ finish in } \Delta t) = \frac{(\mu C\Delta t)^0}{0!}e^{-\mu C\Delta t} = 1-\mu C\Delta t + o\left((\Delta t)^2\right)$$

$$\begin{aligned} P_n(t+\Delta t) &= P_{n-1}(t)\,P(\lambda\Delta t)(1-\mu C\Delta t) + P_{n+1}(t)(1-\lambda\Delta t)(\mu C\Delta t) \\ &\quad + P_n(t)\left[(1-\lambda\Delta t)(1-\mu C\Delta t)+(\lambda\Delta t)(\mu C\Delta t)\right] + o\left((\Delta t)^2\right) \\ &= P_n(t) + \left\{(-\lambda-\mu C)P_n(t) + \lambda P_{n-1}(t) + \mu C P_{n+1}(t)\right\}\Delta t + o\left((\Delta t)^2\right) \end{aligned}$$

Steady State Assumption

For large t, buffer level reaches "steady state" (almost constant)

$B(t) \xrightarrow[t\to\infty]{} n$ and $P_n(t) \xrightarrow[t\to\infty]{} P_n$

New requests balance finished responses

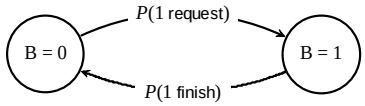
Almost Empty Buffer

$$P\left[B(t)=0 \textbf{ AND } 1 \text{ request}\right] = P\left[B(t)=1 \textbf{ AND } 1 \text{ finish}\right]$$

$$P_0 \times P_{\Delta t}(1 \text{ request}) = P_1 \times P_{\Delta t}(1 \text{ finish})$$

$$P_0 \times \lambda\Delta t = P_1 \times \mu C\Delta t$$

$$P_1 = \frac{\lambda\Delta t}{\mu C\Delta t}P_0 = \frac{\lambda}{\mu C}P_0 = \rho P_0$$



$$\rho = \frac{\lambda}{\mu C} = \frac{\text{average request rate}}{\text{average service rate}} = \textbf{utilization}$$

General Steady State

$P_n(t) \to P_n$ (constant)

$$0 = \frac{d}{dt}P_n(t) = \lim_{\Delta t \to 0}\frac{P_n(t+\Delta t) - P_n(t)}{\Delta t}$$

$$= \lim_{\Delta t \to 0}\frac{\left[(-\lambda-\mu C)P_n(t) + \mu C P_{n+1}(t) + \lambda P_{n-1}(t)\right]\Delta t + o\left((\Delta t)^2\right)}{\Delta t}$$

$$= -P_n(t)(\lambda+\mu C) + \mu C P_{n+1}(t) + \lambda P_{n-1}(t)$$

$$P_{n+1} = \frac{1}{\mu C}\left[(\lambda+\mu C)P_n - \lambda P_{n-1}\right] = \left[\left(\frac{\lambda}{\mu C}+1\right)P_n - \frac{\lambda}{\mu C}P_{n-1}\right]$$

$$= (1+\rho)P_n - \rho P_{n-1}$$

$$\rho = \frac{\lambda}{\mu C}$$

Average Buffer Level

From recursion relation,

$$P_{n+1} = (1+\rho)P_n - \rho P_{n-1}$$

$$P_2 = (1+\rho)P_1 - \rho P_0 = (1+\rho)\rho P_0 - \rho P_0 = \rho^2 P_0$$

$$P_3 = (1+\rho)P_2 - \rho P_1 = (1+\rho)\rho^2 P_0 - \rho P_0 = \rho^3 P_0$$

$$\vdots$$

$$P_n = \rho^n P_0$$

sum of probabilities $= 1 = \sum_{n=0}^{\infty} P_n = \sum_{n=0}^{\infty}\rho^n P_0 = P_0\frac{1}{1-\rho} \Rightarrow \quad P_0 = 1-\rho = 1-\frac{\lambda}{\mu C}$

Probability of Buffer Level n $\quad P_n = \left(\frac{\lambda}{\mu C}\right)^n\left(1-\frac{\lambda}{\mu C}\right) = \rho^n(1-\rho)$

Average Buffer Level $\quad \langle n\rangle = \sum_{n=0}^{\infty} n\rho^n(1-\rho) = \frac{\rho}{1-\rho} = \frac{\lambda/\mu C}{1-\lambda/\mu C} = \frac{\lambda}{\mu C-\lambda}$

Total Latency and Buffer Level

$$\langle T\rangle = \langle\text{service time for one request}\rangle + \langle\text{buffer latency}\rangle$$

$$= \langle\text{service time for one request}\rangle + \langle\text{time to service FIFO queue}\rangle$$

$$= \frac{1}{\mu C} + \langle n\rangle\frac{1}{\mu C} = \frac{1}{\mu C} + \left[\frac{\lambda/\mu C}{1-\lambda/\mu C}\right]\frac{1}{\mu C}$$

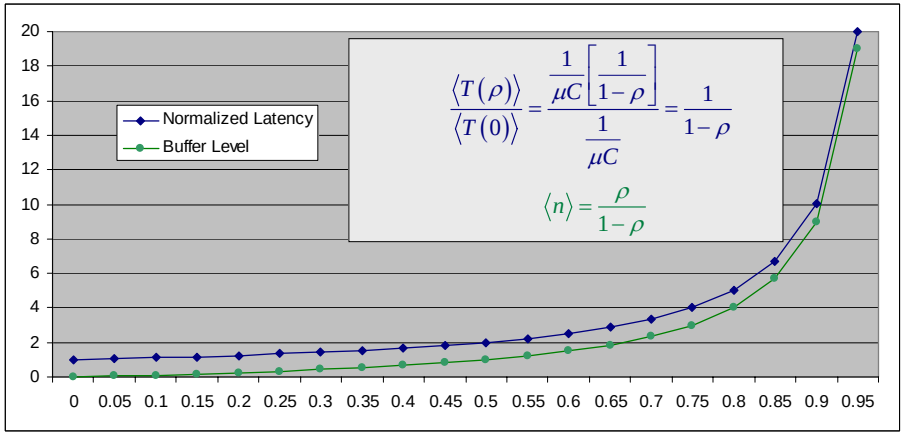
$$= \frac{1}{\mu C}\left[1+\frac{\lambda/\mu C}{1-\lambda/\mu C}\right] = \frac{1}{\mu C}\left[\frac{1}{1-\lambda/\mu C}\right] = \frac{1}{\mu C}\left[\frac{1}{1-\rho}\right]$$

$$= \frac{1}{\mu C-\lambda} = \frac{1}{\text{average service rate} - \text{average request rate}}$$

Little's Law

$$\langle n\rangle = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu C}\left[\frac{1}{1-\rho}\right] = \lambda\cdot\langle T\rangle = (\text{average request rate})\times(\text{average latency})$$

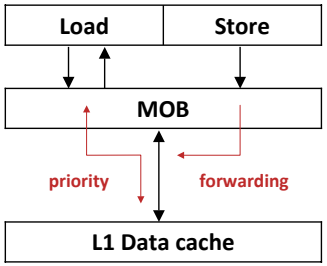
Latency and Buffer Level versus Utilization



$$\text{Normalized Latency} = \frac{\text{Latency (seconds)}}{\text{Latency with zero utilization}}$$

Memory Queuing

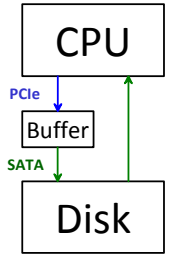
Memory Ordering Buffer (MOB)



- Between Load/Store EU and data cache
  - Buffers read/write operations
  - Enables "simultaneous" reads and writes
- Issues loads before stores
  - Enhances throughput of speculative reads
- Forwards queued stores to read requests

Hard drive

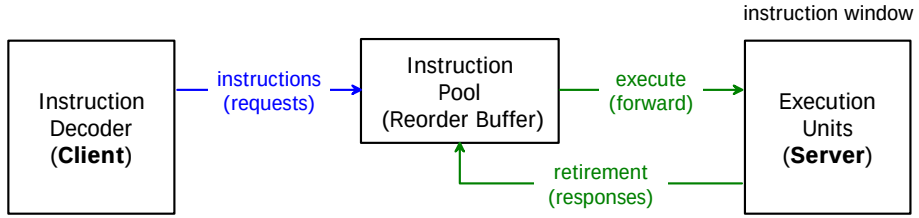
- SATA 3.0 sustained throughput = 6 Gbps < 600 MB/s
- PCI Express 4.0 sustained throughput < 16 GB/sec
- Typical disk write buffer = 64 MB
- Overflow time = 64 MB / [(16 – 0.6) GB/s] = 4.16 ms



Safe removal of external drive

- Dump buffer to file
- Mark file closed in file system
- Unsafe removal ⇒ possible data loss

Client/Server Model of CPU



Client

Decoder sends λ instructions per cycle to ROB

Server

EUs can operate on μC instructions (one instruction window) per cycle

Goal

Maximize utilization of EUs $\rho = \frac{\lambda}{\mu C}$ without ROB overflow

For fixed instruction window μC $\Rightarrow$ maximize λ

No overflow $\Rightarrow$ average decoder rate < instruction window

General Superscalar Model

Execution units (EUs) operate in parallel

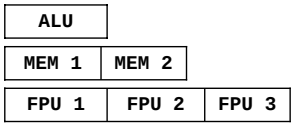
EU stages ≥ 1

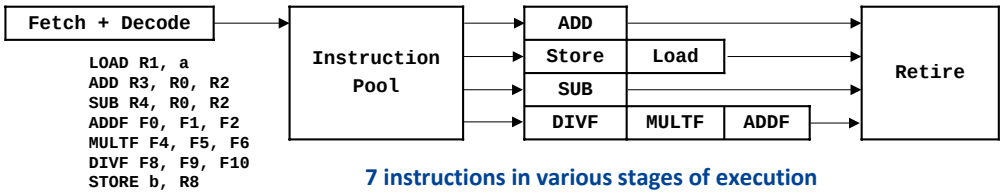
Ideal case

- Every stage of every EU working on every clock cycle
- Multiple instructions pipelined through EU stages

Example

- 2 ALUs — 1 cycle per instruction
- 1 Load + Store — 2 cycles per instruction
- 1 FPU — 3 cycles per instruction





7 instructions in various stages of execution

Detailed Analysis of ILP

Pipeline structure

u_i = execution units (EU) of type i

$u = \sum_i u_i$ = total execution units (EU) in CPU

s_i = pipeline stages in EU of type i

$u_i \times s_i$ = pipeline stages of type i

$IC_{EU} = \sum_i u_i \times s_i$ = total pipeline stages in CPU

= instructions executing in all EUs

= size of instruction window

= instructions executing in parallel (ILP)

$$s = \sum_i \frac{u_i}{u} \times s_i = \frac{\sum_i u_i \times s_i}{u} = \frac{IC_{EU}}{u} = \text{average pipeline stages in EUs}$$

instruction window $= IC_{EU} = u \times s$

Queuing Theory and Decoder Rate

Instructions Per Clock Cycle

$$IPC_i = \frac{1}{CPI_i} = u_i \times IPC_i^{scalar} = \frac{u_i}{CPI_i^{scalar}} = \frac{u_i}{1 + CPI_i^{stall}}$$

$$IPC = \sum_i IPC_i = \sum_i \frac{u_i}{1 + CPI_i^{stall}} = \mu C = \text{average response rate}$$

$$\text{Average buffer (ROB) level} = \langle n \rangle = \frac{\lambda}{\mu C - \lambda} = \frac{\lambda}{IPC - \lambda}$$

Maximize utilization $\Rightarrow$ ROB level = instruction window

$$\langle n \rangle = \frac{\lambda}{IPC - \lambda} = IC_{EU} \Rightarrow \lambda = \frac{IC_{EU} \times IPC}{1 + IC_{EU}} = \frac{s \times u \times IPC}{1 + s \times u}$$

$$CPI^{stall} = 0 \Rightarrow IPC^{ideal} = \sum_i u_i = u$$

$$\lambda^{ideal} = \frac{s \times u^2}{1 + s \times u} = \text{ instructions attempting to execute per cycle}$$

Example of Instruction Window

Execution units

2 integer ALUs — 1 CC per execution
2 FPUs — 3 CC per execution
2 load/store units — 2 CC per execution

Instruction window

$$u = \sum_i u_i = 6 = IPC^{ideal}$$

$$s = \sum_i \frac{u_i}{u} \times s_i = \frac{2}{6} \times 1 + \frac{2}{6} \times 3 + \frac{2}{6} \times 2 = \frac{12}{6} = 2$$

$IC_{EU} = u \times s = 12$ instructions executing in parallel

Decoder rate (ideal)

Maximize $\rho = \frac{\lambda}{\mu C} \Rightarrow \lambda = \lambda^{ideal}$ without overflow (ROB level = IC_{EU})

$$IPC^{ideal} = \mu C = 6 > \lambda = \lambda^{ideal} = \frac{s \times u^2}{1 + s \times u} = \frac{12 \times 6}{1 + 12} = 5.54$$

Implications for ILP Scalability

Scaling instruction window and decoder rate

execution units $u_i \rightarrow u_i' = \alpha_u u_i$
pipeline stages $s_i \rightarrow s_i' = \beta_s s_i$
instruction window $IC_{EU} \rightarrow IC_{EU}' = \alpha_u \beta_s IC_{EU}$

$$\lambda^{ideal} \rightarrow \lambda^{ideal}{}' = \frac{\left(\beta_s \alpha_u^2\right) s \times u^2}{1 + \left(\beta_s \alpha_u\right) s \times u}$$

Scaling 6 $\rightarrow$ 15 EUs with 2 $\rightarrow$ 8 superpipelined stages

$$\alpha_u = \frac{15}{6} \quad \beta_s = \frac{8}{2} \quad \Rightarrow \quad \alpha_u \times \beta_s = 10$$

$IC_{EU} = 120$ instructions executing in parallel

$15 > \lambda^{ideal} \geq 14.9$ instructions decoded per CC

Difficulties

Decode 15 instructions per CC with cache misses, mispredictions, ...
Maintain window of 120 independent instructions
25 – 30 branch instructions in window $\Rightarrow$ large misprediction probability