System Design for a Million TPS

Hüsnü Sensoy
Global Maksimum Data & Information Technologies

Global Maksimum Data & Information Technologies

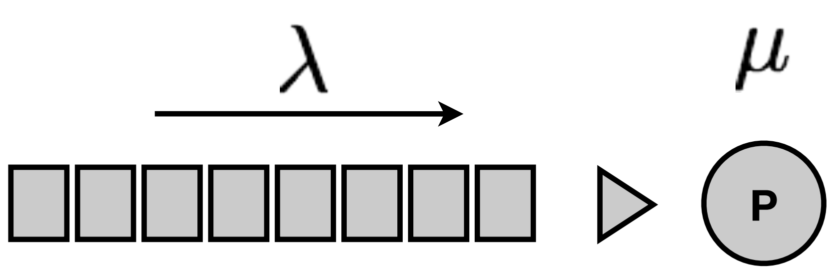
- Focused just on large scale data and information problems.
- Complex Event Processing
  - Oracle CEP
  - Making 500 different business decisions for 1.2 Millions of events in a second
- Data Mining
  - Oracle Data Mining
- Large scale data analytics
  - Ten billion rows in a week

Agenda: How to Design Systems for Extreme Performance

- Latency and Throughput
- Know your Hardware
- Physics rules it !!!
- Test it. But really test it

Latency and Throughput

- People usually use following statements as if they mean the same thing:
  - A low latency application.
  - We need to process many events per second.



$$Latency_{Average} = \frac{1}{(\mu - \lambda)}$$

$$QueueLengthNeeded_{MIN} = \frac{\lambda}{(\mu - \lambda)}$$

$$Throughput \equiv \lambda$$

$$Throughput_{MAX} = \mu$$

Inference

$$Latency_{Average} = \frac{1}{(\mu - \lambda)}$$

$$QueueLengthNeeded_{MIN} = \frac{\lambda}{(\mu - \lambda)}$$

$$Throughput \equiv \lambda$$

$$Throughput_{MAX} = \mu$$

$$Latency_{Average} = \frac{1}{(\mu - \lambda)}$$

$$QueueLengthNeeded_{MIN} = \frac{\lambda}{(\mu - \lambda)}$$

$$Throughput \equiv \lambda$$

$$Throughput_{MAX} = \mu$$

Inference

- If some application asks for 30 MB/s from disk IO subsystem your throughput is 30 MB/s and this does not tell any thing about the latency per MB.

$Latency_{Average} = \frac{1}{(\mu - \lambda)}$

$QueueLengthNeeded_{MIN} = \frac{\lambda}{(\mu - \lambda)}$

$Throughput \equiv \lambda$

$Throughput_{MAX} = \mu$

Inference

- If some application asks for 30 MB/s from disk IO subsystem your throughput is 30 MB/s and this does not tell any thing about the latency per MB.

- If the disk IO system requires 5 msec per 1 MB request, maximum throughput is 200 MB/s

$$Latency_{Average} = \frac{1}{(\mu - \lambda)}$$

$$QueueLengthNeeded_{MIN} = \frac{\lambda}{(\mu - \lambda)}$$

$$Throughput \equiv \lambda$$

$$Throughput_{MAX} = \mu$$

Inference

- If some application asks for 30 MB/s from disk IO subsystem your throughput is 30 MB/s and this does not tell any thing about the latency per MB.
- If the disk IO system requires 5 msec per 1 MB request, maximum throughput is 200 MB/s
- If the application asks for 60 MB/s from disk IO subsystem with a maximum throughput of 200 MB/s, each 1 MB will be served in 7 msec on the average

GLOBAL MAKSIMUM

$$Latency_{Average} = \frac{1}{(\mu - \lambda)}$$

$$QueueLengthNeeded_{MIN} = \frac{\lambda}{(\mu - \lambda)}$$

$$Throughput \equiv \lambda$$

$$Throughput_{MAX} = \mu$$

Inference

- If some application asks for 30 MB/s from disk IO subsystem your throughput is 30 MB/s and this does not tell any thing about the latency per MB.
- If the disk IO system requires 5 msec per 1 MB request, maximum throughput is 200 MB/s
- If the application asks for 60 MB/s from disk IO subsystem with a maximum throughput of 200 MB/s, each 1 MB will be served in 7 msec on the average
- If the application asks for 150 MB/s from disk IO subsystem with a maximum throughput of 200 MB/s, each 1 MB will be served in 20 msec on the average

GLOBAL MAKSIMUM

$$Latency_{Average} = \frac{1}{(\mu - \lambda)}$$

$$QueueLengthNeeded_{MIN} = \frac{\lambda}{(\mu - \lambda)}$$

$$Throughput \equiv \lambda$$

$$Throughput_{MAX} = \mu$$

Inference

- If some application asks for 30 MB/s from disk IO subsystem your throughput is 30 MB/s and this does not tell any thing about the latency per MB.
- If the disk IO system requires 5 msec per 1 MB request, maximum throughput is 200 MB/s
- If the application asks for 60 MB/s from disk IO subsystem with a maximum throughput of 200 MB/s, each 1 MB will be served in 7 msec on the average
- If the application asks for 150 MB/s from disk IO subsystem with a maximum throughput of 200 MB/s, each 1 MB will be served in 20 msec on the average
- For throughput demanding applications increasing demand rate will simply increase the throughput as long as you have sufficiently long request queue.

$$Latency_{Average} = \frac{1}{(\mu - \lambda)}$$

$$QueueLengthNeeded_{MIN} = \frac{\lambda}{(\mu - \lambda)}$$

$$Throughput \equiv \lambda$$

$$Throughput_{MAX} = \mu$$

Inference

- If some application asks for 30 MB/s from disk IO subsystem your throughput is 30 MB/s and this does not tell any thing about the latency per MB.
- If the disk IO system requires 5 msec per 1 MB request, maximum throughput is 200 MB/s
- If the application asks for 60 MB/s from disk IO subsystem with a maximum throughput of 200 MB/s, each 1 MB will be served in 7 msec on the average
- If the application asks for 150 MB/s from disk IO subsystem with a maximum throughput of 200 MB/s, each 1 MB will be served in 20 msec on the average
- For throughput demanding applications increasing demand rate will simply increase the throughput as long as you have sufficiently long request queue.
- For real-time applications this is not true. You can not have arbitrarily long queues. Because all real-time applications have a scheduling deadline.
    - As opposed to common thinking, real-time is not about being fast but about being deterministic.

Inference

$Latency_{Average} = \frac{1}{(\mu - \lambda)}$

$QueueLengthNeeded_{MIN} = \frac{\lambda}{(\mu - \lambda)}$

$Throughput \equiv \lambda$

$Throughput_{MAX} = \mu$

- If some application asks for 30 MB/s from disk IO subsystem your throughput is 30 MB/s and this does not tell any thing about the latency per MB.
- If the disk IO system requires 5 msec per 1 MB request, maximum throughput is 200 MB/s
- If the application asks for 60 MB/s from disk IO subsystem with a maximum throughput of 200 MB/s, each 1 MB will be served in 7 msec on the average
- If the application asks for 150 MB/s from disk IO subsystem with a maximum throughput of 200 MB/s, each 1 MB will be served in 20 msec on the average
- For throughput demanding applications increasing demand rate will simply increase the throughput as long as you have sufficiently long request queue.
- For real-time applications this is not true. You can not have arbitrarily long queues. Because all real-time applications have a scheduling deadline.
    - As opposed to common thinking, real-time is not about being fast but about being deterministic.
- For real-time applications what you can do it to make processor(s) faster or reject the requests if possible.

GLOBAL MAKSIMUM

$$Latency_{Average} = \frac{1}{(\mu - \lambda)}$$

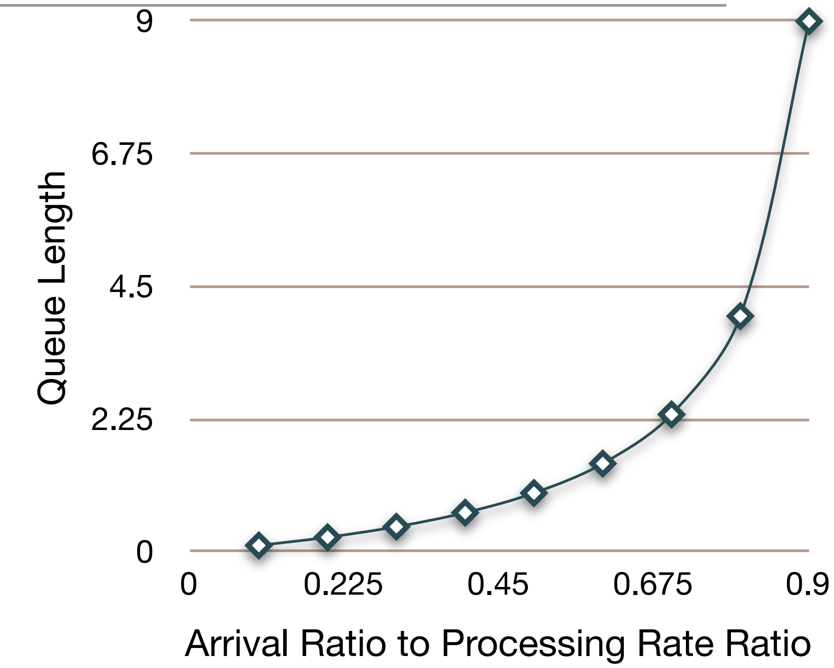
$$QueueLengthNeeded_{MIN} = \frac{\lambda}{(\mu - \lambda)}$$

$$Throughput \equiv \lambda$$

$$Throughput_{MAX} = \mu$$

Inference

- If some application asks for 30 MB/s from disk IO subsystem your throughput is 30 MB/s and this does not tell any thing about the latency per MB.
- If the disk IO system requires 5 msec per 1 MB request, maximum throughput is 200 MB/s
- If the application asks for 60 MB/s from disk IO subsystem with a maximum throughput of 200 MB/s, each 1 MB will be served in 7 msec on the average
- If the application asks for 150 MB/s from disk IO subsystem with a maximum throughput of 200 MB/s, each 1 MB will be served in 20 msec on the average
- For throughput demanding applications increasing demand rate will simply increase the throughput as long as you have sufficiently long request queue.
- For real-time applications this is not true. You can not have arbitrarily long queues. Because all real-time applications have a scheduling deadline.
    - As opposed to common thinking, real-time is not about being fast but about being deterministic.
- For real-time applications what you can do it to make processor(s) faster or reject the requests if possible.



Inference

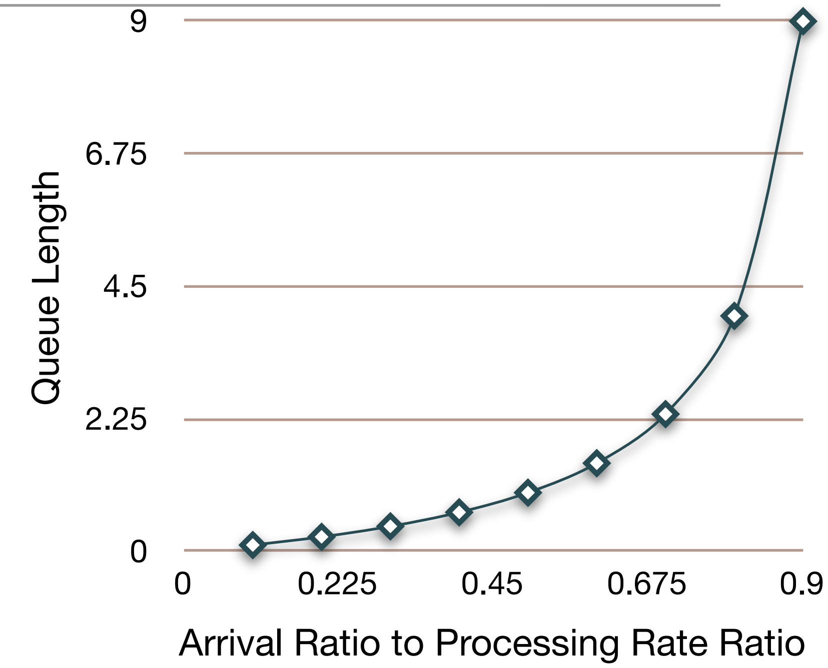
$$Latency_{Average} = \frac{1}{(\mu-\lambda)}$$

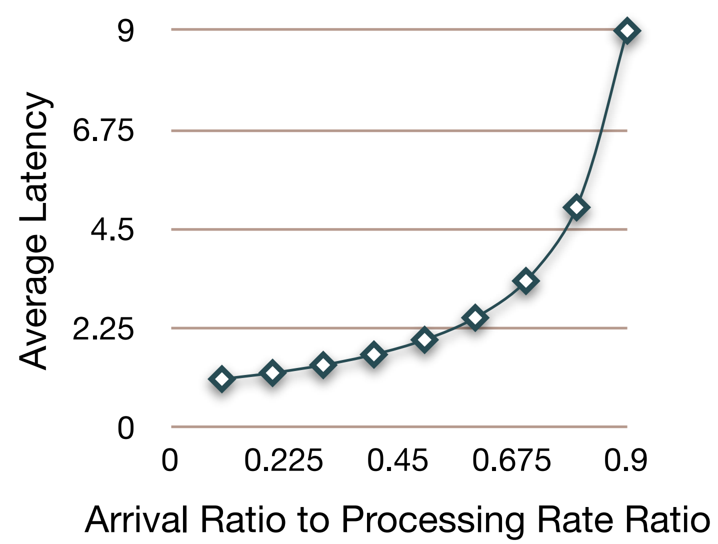
$$QueueLengthNeeded_{MIN} = \frac{\lambda}{(\mu-\lambda)}$$

$$Throughput \equiv \lambda$$

$$Throughput_{MAX} = \mu$$

- If some application asks for 30 MB/s from disk IO subsystem your throughput is 30 MB/s and this does not tell any thing about the latency per MB.
- If the disk IO system requires 5 msec per 1 MB request, maximum throughput is 200 MB/s
- If the application asks for 60 MB/s from disk IO subsystem with a maximum throughput of 200 MB/s, each 1 MB will be served in 7 msec on the average
- If the application asks for 150 MB/s from disk IO subsystem with a maximum throughput of 200 MB/s, each 1 MB will be served in 20 msec on the average
- For throughput demanding applications increasing demand rate will simply increase the throughput as long as you have sufficiently long request queue.
- For real-time applications this is not true. You can not have arbitrarily long queues. Because all real-time applications have a scheduling deadline.
    - As opposed to common thinking, real-time is not about being fast but about being deterministic.
- For real-time applications what you can do it to make processor(s) faster or reject the requests if possible.





Choose Your Hardware

Choose Your Hardware

- Do you need a higher frequency or more cores ?
    - Intel follows the strategy to inversely correlate number of cores and per core frequency.

GLOBAL MAKSIMUM

Choose Your Hardware

- Do you need a higher frequency or more cores ?
  - Intel follows the strategy to inversely correlate number of cores and per core frequency.
- Is your application designed for scale up or scale out ?

Choose Your Hardware

- Do you need a higher frequency or more cores ?
  - Intel follows the strategy to inversely correlate number of cores and per core frequency.
- Is your application designed for scale up or scale out ?
- Is your application NUMA aware ?

Choose Your Hardware

- Do you need a higher frequency or more cores ?
  - Intel follows the strategy to inversely correlate number of cores and per core frequency.
- Is your application designed for scale up or scale out ?
- Is your application NUMA aware ?
- Cache Configuration
  - L1/L2/L3 Size
  - Cache block unit size

Choose Your Hardware

- Do you need a higher frequency or more cores ?
  - Intel follows the strategy to inversely correlate number of cores and per core frequency.
- Is your application designed for scale up or scale out ?
- Is your application NUMA aware ?
- Cache Configuration
  - L1/L2/L3 Size
  - Cache block unit size
- Do you need more memory or do you need fast memory ?

GLOBAL MAKSIMUM

Save Money on Hardware

Save Money on Hardware

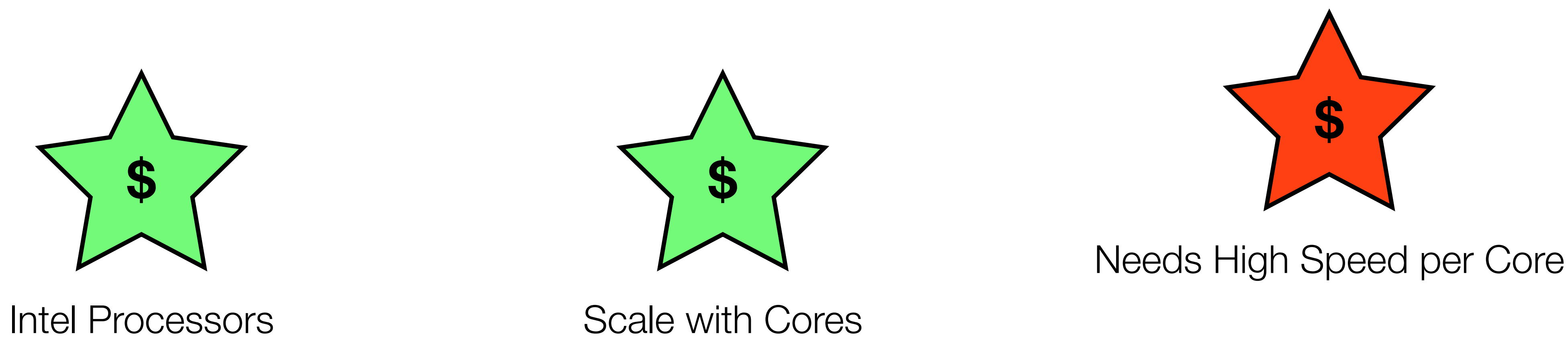
Intel Processors

Save Money on Hardware

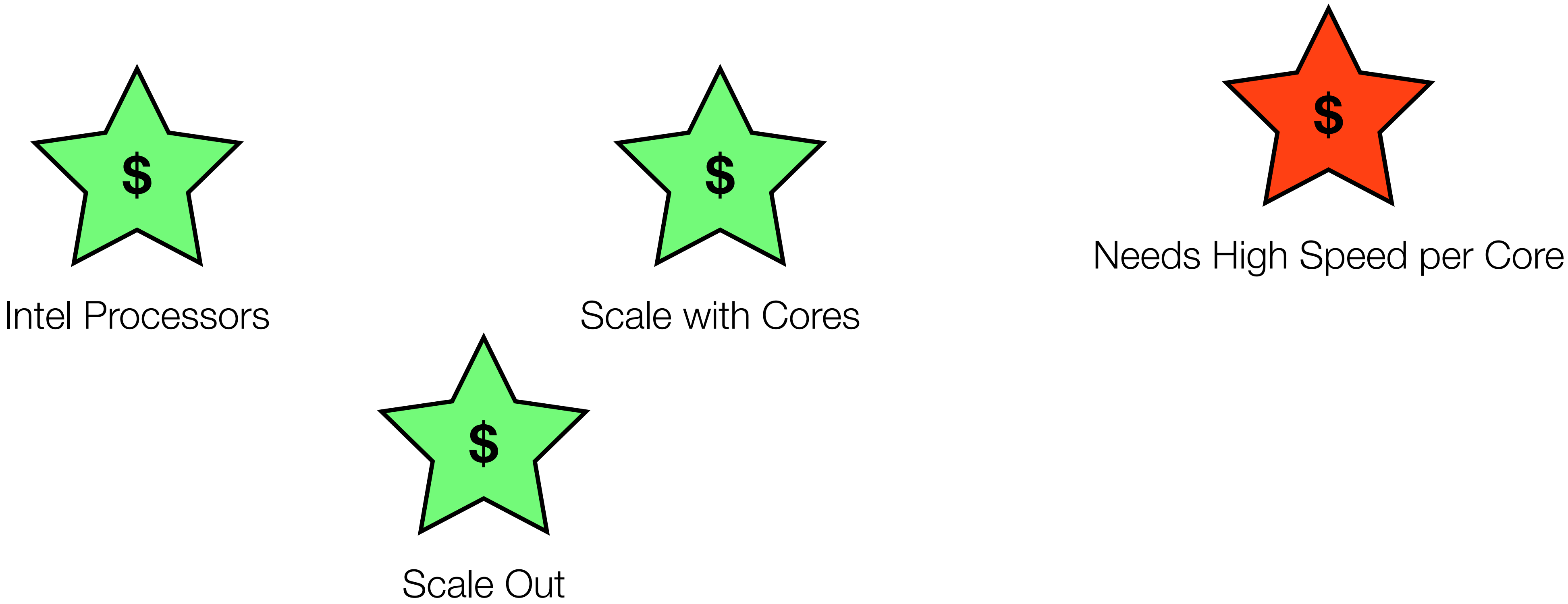
Intel Processors

Needs High Speed per Core

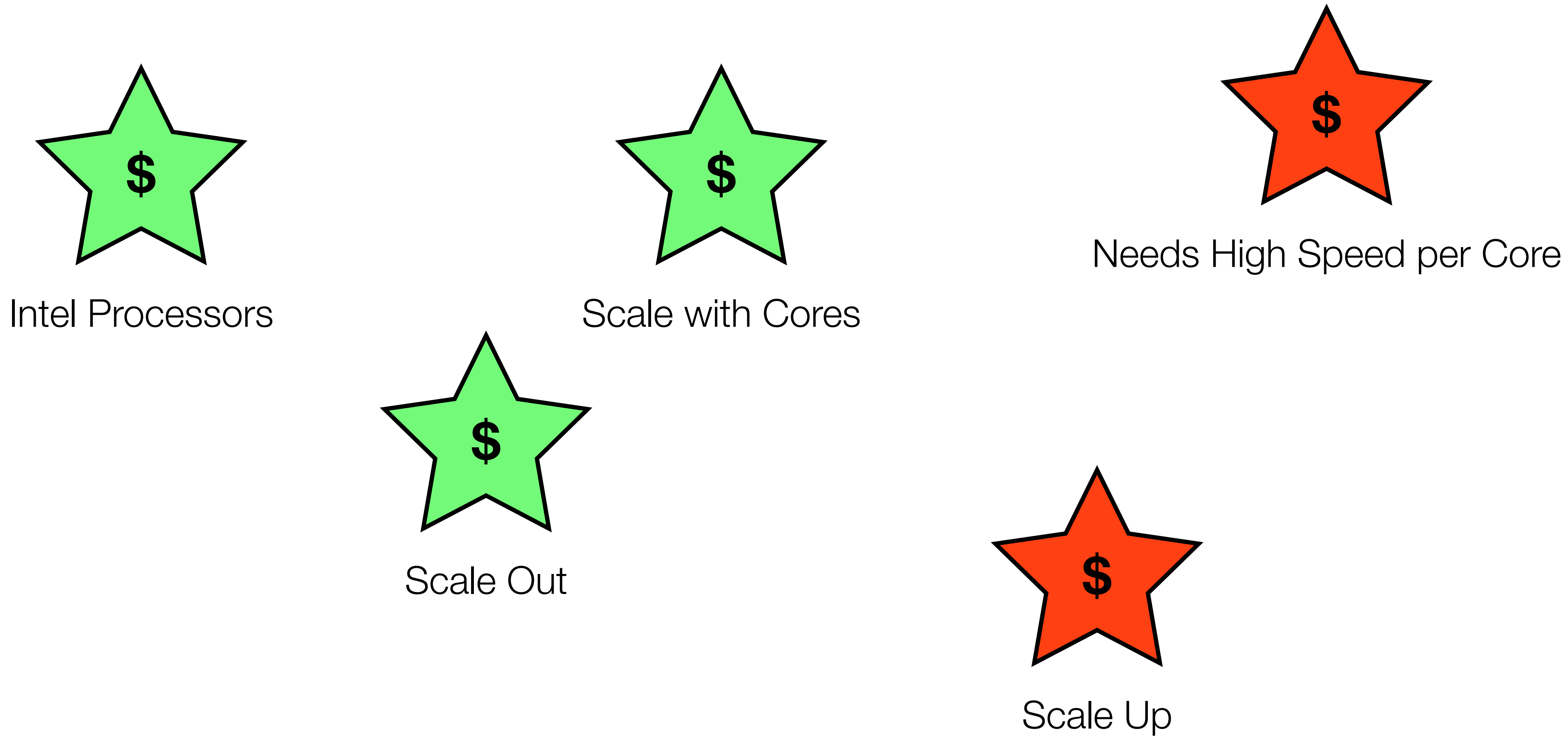
Save Money on Hardware



Save Money on Hardware



Save Money on Hardware



Know Your Hardware

- What extensions does your processor support ?
  - SSE, MMX, AVX, ...
- Your server is running in max performance/power efficient/balanced-mode ?
- What is the maximum PCI but speed allowed by your configuration ?

GLOBAL MAKSIMUM

Physics Rule It

Physics Rule It

- No matter how fast your single box is, it has a limit
    - Even you have a 8 socket machine

Physics Rule It

- No matter how fast your single box is, it has a limit
    - Even you have a 8 socket machine
- In order to go faster you need to put multiple boxes connected to each other.

Physics Rule It

- No matter how fast your single box is, it has a limit
  - Even you have a 8 socket machine
- In order to go faster you need to put multiple boxes connected to each other.
- This connections must be
  - Fast
    - 2.5 GB/s @ 32K block
    - 9 micro second @ 2K block SDP
    - 30 micro second @ 2K block TCP
  - Reliable
    - Bonding

Scale Out

- Scaling out an application requires application to be written scale-out in mind no matter how fast your inter connect fabric is.
- Otherwise you will never achieve linearity.
- Key word in hear is HASHing

Real Testing

- Many development teams do not test their applications for
  - Performance
  - Availability
- One reason is the cost of putting an identical test system configuration.
- If you can seriously reduce the hardware cost you can create an identical test system. So that you can perform real tests.

Stop Swapping for Real-Time

- If your application needs large amount of memory allocation then there is always a risk for OS to swap out your memory page to disk until you need it.
- Many applications on Linux (Oracle Database, Java JVM, etc) supports allocating memory from Huge Page pool.
- Huge Pages are pinned into memory and skipped by Linux swap process ensuring that they will be in physical memory whenever you refer to them.
- Moreover since Huge Pages are larger than standard ones, OS spends less CPU cycles on TLB

50% = 100%

- For some applications you may observe a maximum server utilization of 50%.
- This may be related to the CPU IO balance of your application running on server.
- If your application really hammers on CPU there is no way to utilize servers at 100%

Conclusion

- Define your application scope in terms of
  - Performance
  - Availability
  - Scalability
- Choose your hardware appropriately
- Use your hardware appropriately
- Hardware & Configuration is only 40% of it. Rest is all about the software you implement on it.

Q & A

GLOBAL MAKSIMUM

Don’t stuck at Local Minima(s)...

husnu.sensoy@globalmaksimum.com

husnu.sensoy@gmail.com

http://husnusensoy.wordpress.com

@husnusensoy