
GLOC
Great Lakes Oracle Conference
Understanding SPARC Processor Performance
MAY 15 & 16, 2019
CLEVELAND PUBLIC AUDITORIUM, CLEVELAND, OHIO
WWW.NEOOUG.ORG/GLOC

About the Speaker

- Akiva Lichtner
- Physics background
- Twenty years experience in IT
- Enterprise production support analyst
- Java developer
- Oracle query plan manager …
- Spoke here at G.L.O.C. about TDD and Java dynamic tracing

Audience

- Developers
- System administrators
- Tech support analysts
- IT managers

Motivation

- I have been working in tech support for a large application
- We have run SPARC T4 servers and now we run T7 servers
- Application servers, database servers
- Environments are all different
- Users complained for years about “environment X” being slow, finally figured out why
- What I learned can be very useful for users of SPARC servers

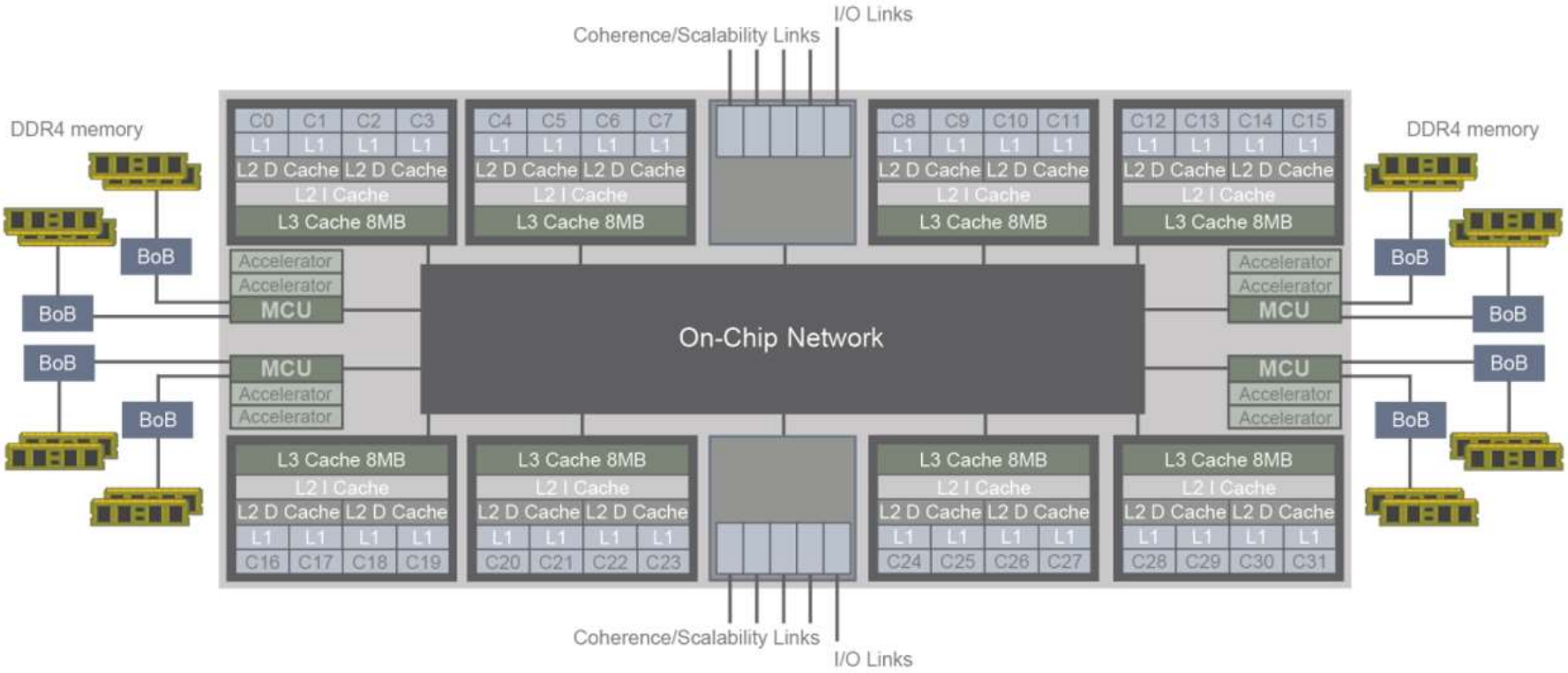
What is SPARC?

- First released in 1987, created by Sun Microsystems to replace the Motorola 68000 in its workstation products
- During the .com boom Solaris/SPARC and Windows/Intel were the only supported platforms for the JVM
- In 2000 the bubble burst and Sun server sales plunged
- Sun acquired Afara Websystems, which had built an interesting new processor, and renamed it the UltraSPARC T1
- Was followed by T2 through M8, evolutions of the same design
- More recently Oracle has added significant new functionality

Processor Design

- High core count (even in the early days)
- Many threads per core
- “Barrel” processor
- Designed to switch efficiently
- Non-uniform memory access
- Per-processor shared cache
- Core-level shared cache

A picture speaks a thousand words …



What happens if my threads run on the same core?

- Core runs one thread at a time
- Context switch is instantaneous
- Ideally memory latency hides context switch entirely
- However the caches have a limited size
- Performance will depend on cache utilization
- Are the threads on the same core “related” or not?
- I had a CPU-intensive application that would slow down by a factor of five at random times, in the presence of another CPU-intensive application – coincidence?
- To Solaris one core looks like 8 virtual processors

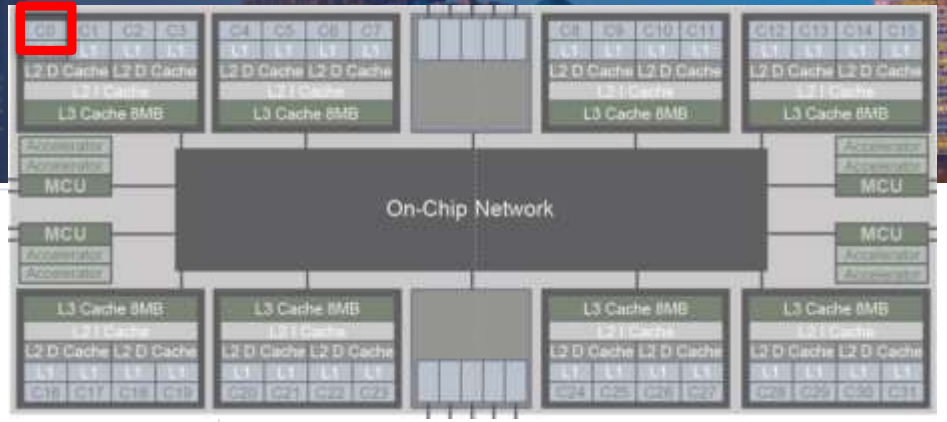
What performance should I expect in general?

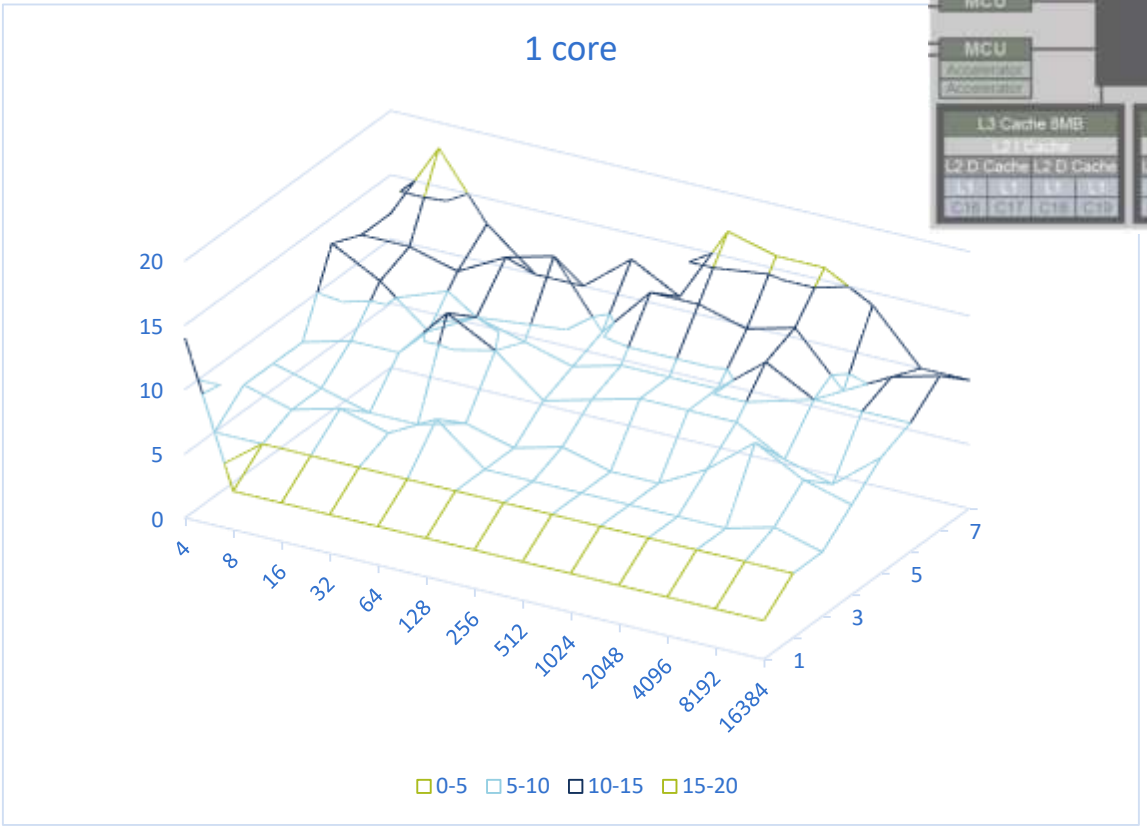
- It all depends
- Common for a Java application to have hundreds of threads
- These servers often consolidate *independent* applications
- By default, Solaris uses *all* the virtual processors
- *Not* a massive SMP machine
- *Not* a traditional CC-NUMA machine

Some experiments to illustrate

- One Java process, N threads
- Random reads/writes on one array of data
- Used pbind to influence choice of virtual processors

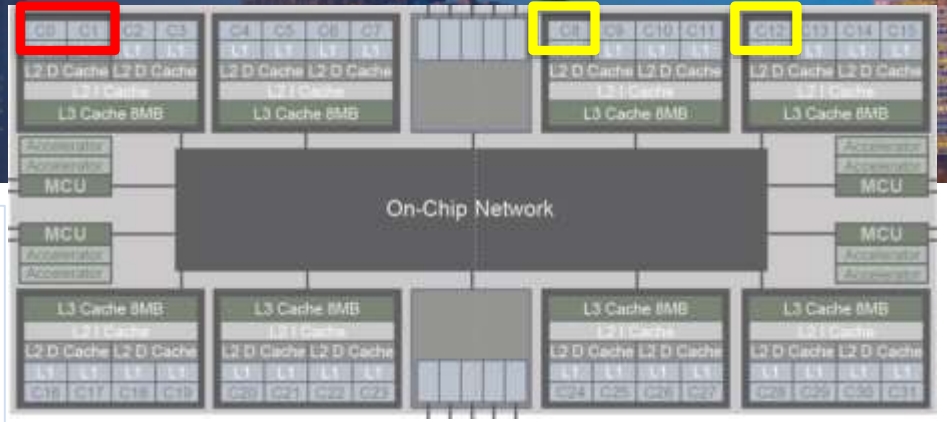
One core



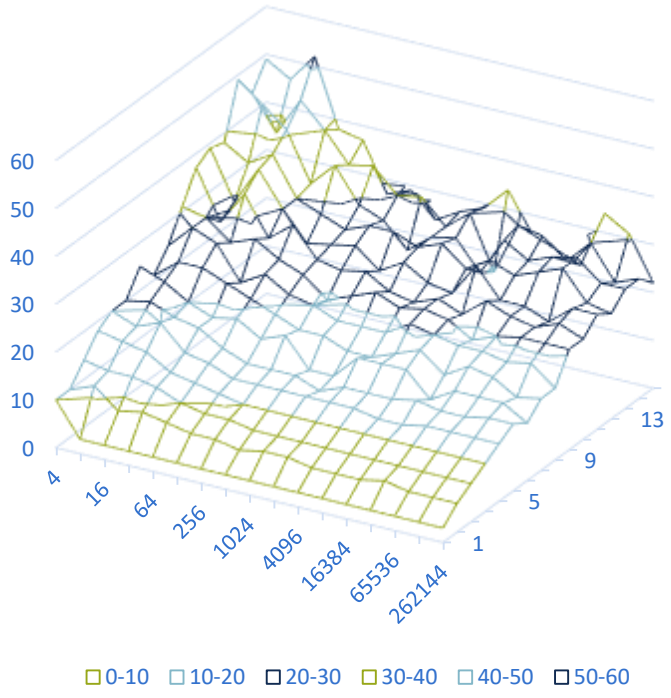


GLOC
Great Lakes Oracle Conference

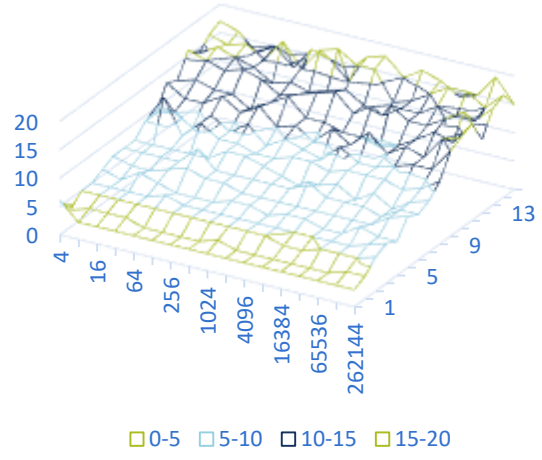
Two cores



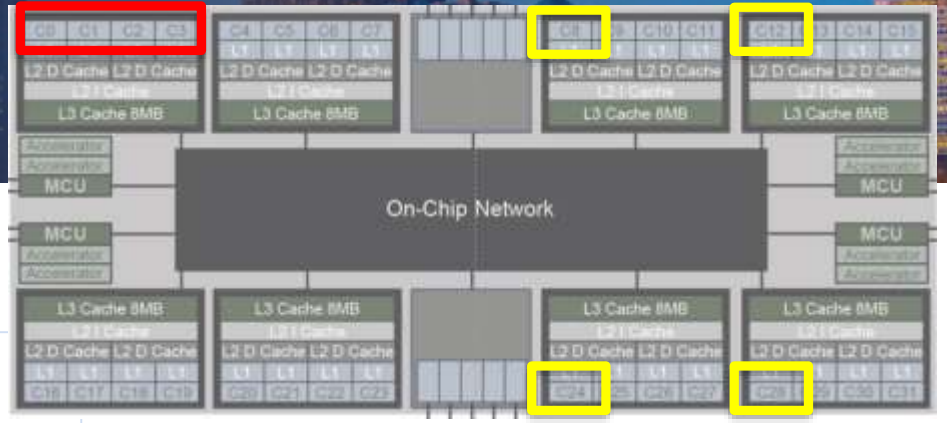
2 non-adjacent cores



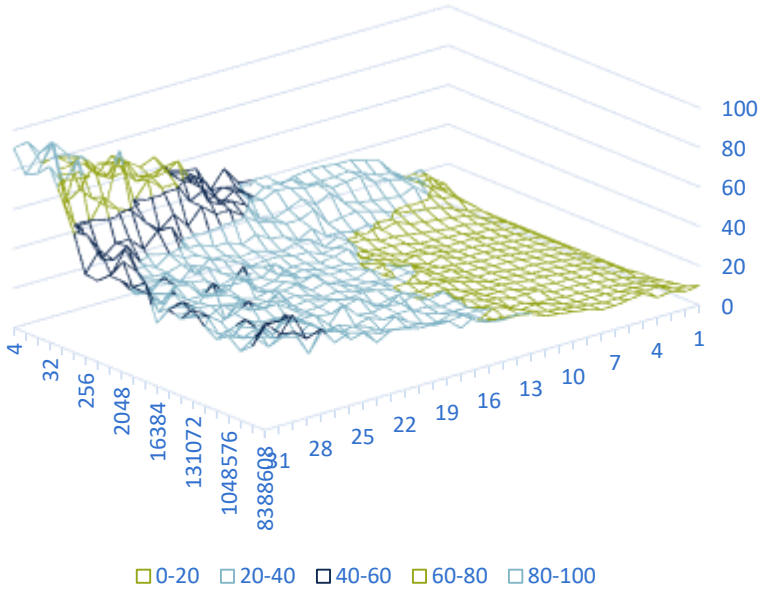
2 adjacent cores



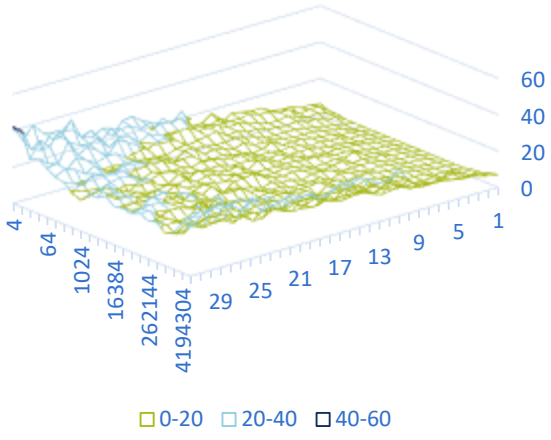
Four cores



4 non-adjacent cores



4 adjacent cores



Multi-processor server

- For several years I had a problematic server
- I finally noticed that it had two processors
- Applications were mostly idle, but load average was high
- I think thread migration overhead was inflating the run queue
- A simple Java memory allocation test was 60% slower than on a single-processor server
- Once a single-processor server was installed, the problem disappeared

Choose application software carefully

- Well-designed concurrent applications
- Learn good concurrent programming
- Streaming applications
  - Each core has a Data Analytics Acceleration (DAX) pipeline
  - compare / filter / translate / compress
  - Oracle Database, Apache Spark query acceleration
  - Java Streams API
- Neural networks?

Solaris Resource Management

- Also known as Solaris Zones
- Not virtualization, still just one server
- Solaris “projects”
- Assign application processes to projects
- Create (virtual) processor sets
- Allocate virtual processors to processor sets

Solaris VM Server for SPARC

- Also known as Logical Domains
- This is virtualization
- Hypervisor
- Multiple instances of Solaris on the same server
- Allocate processors to different VMs

Dynamic Hardware Domains

- Hardware feature in M-Series servers
- Can add/delete/replace processor boards to domains

Take-aways

- There’s a lot to a modern SPARC processor
- This is not an SMP, not a CC-NUMA
- Complex, hierarchical structure
- With great power comes great responsibility
- If performance is a problem, consider your workload sharing
- Keep different applications on different cores
- Try to keep related apps on adjacent cores
- VMs and Zones are not nice to have, you need them