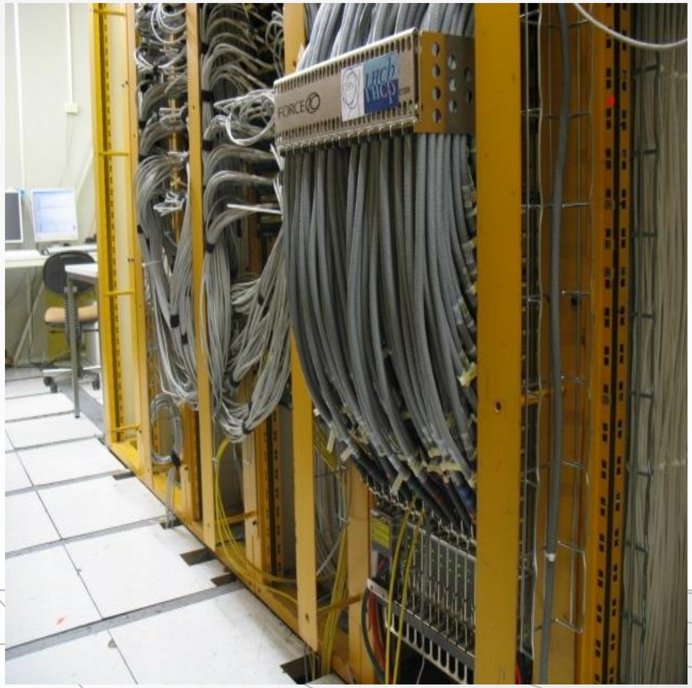
# Performance benchmark of LHCb code on state-of-the-art x86 architectures

Daniel Hugo Campora Perez, Niko Neufled, Rainer Schwemmer

CHEP 2015 - Okinawa

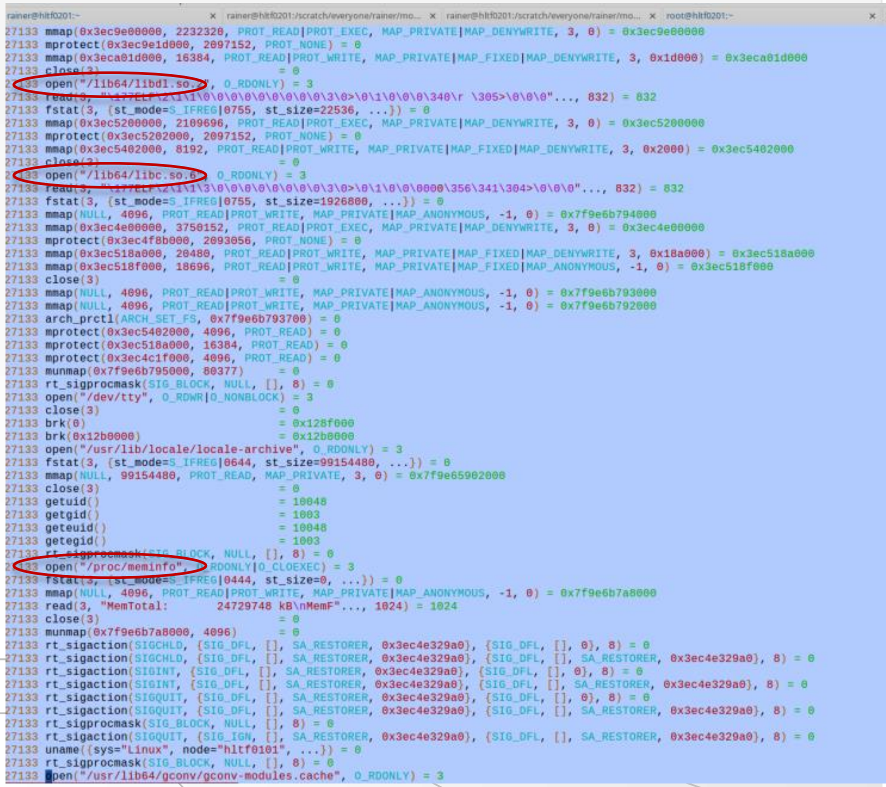
# Background

- The LHCb High Level Trigger (HLT) farm
  - 1800 servers
  - Approximately 40.000 x86 compatible cores
  - Reduce detector data stream from 50-60 GB/s to 1 GB/s
  - 100 m underground
  - Heterogeneous compute cluster
- Farm Upgrade
  - More computing power is necessary to handle the LHC lumi upgrade
  - Decommissioned 550 oldest machines (Nehalem Based Systems)
  - Bought 800 Haswell systems as replacement
- Constraints
  - Farm needs to be able to filter an additional 700.000 events per second (2/3 of old capacity)
  - Has to fit in a 200 kW power envelope due to cooling limitations



# Benchmark

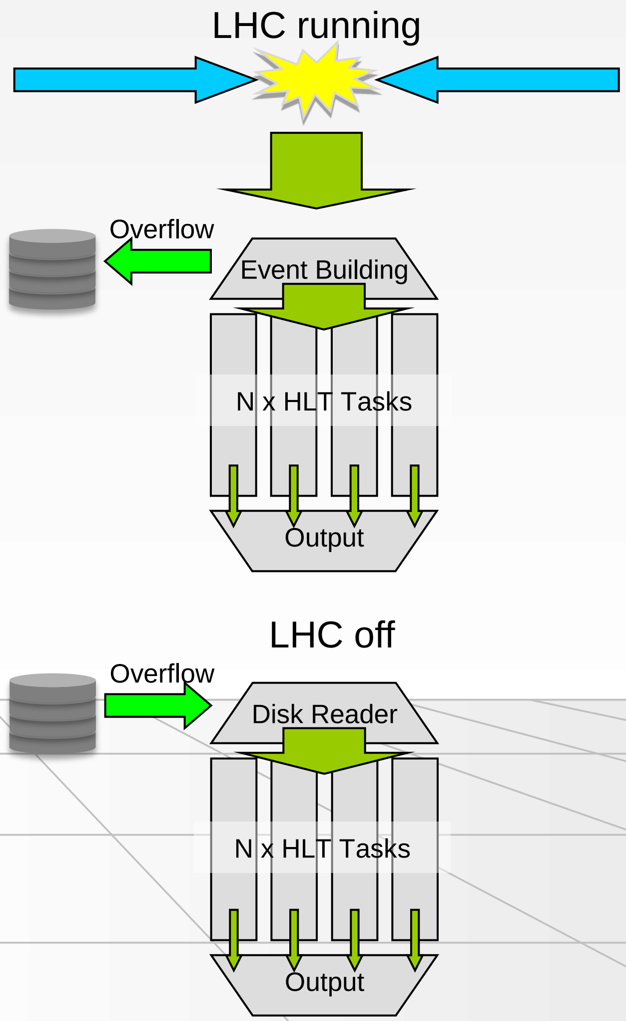
- Determine the most cost efficient compute platform
- Help our colleagues to tune the Trigger
- Create a live DVD to distribute to vendors
- DVD has only 4 GB of space
  - 1.x GB of operating system
  - 2 GB of sample data for input
  - Leaves < 1 GB of space for actual HLT software
  - Need to somehow get rid of unnecessary baggage in our software packages
- Use “strace” on running HLT instance to find the minimum number of files required for the trigger
- Copy onto live DVD
  - + One touch wrapper script



Capture every: open/stat/execv

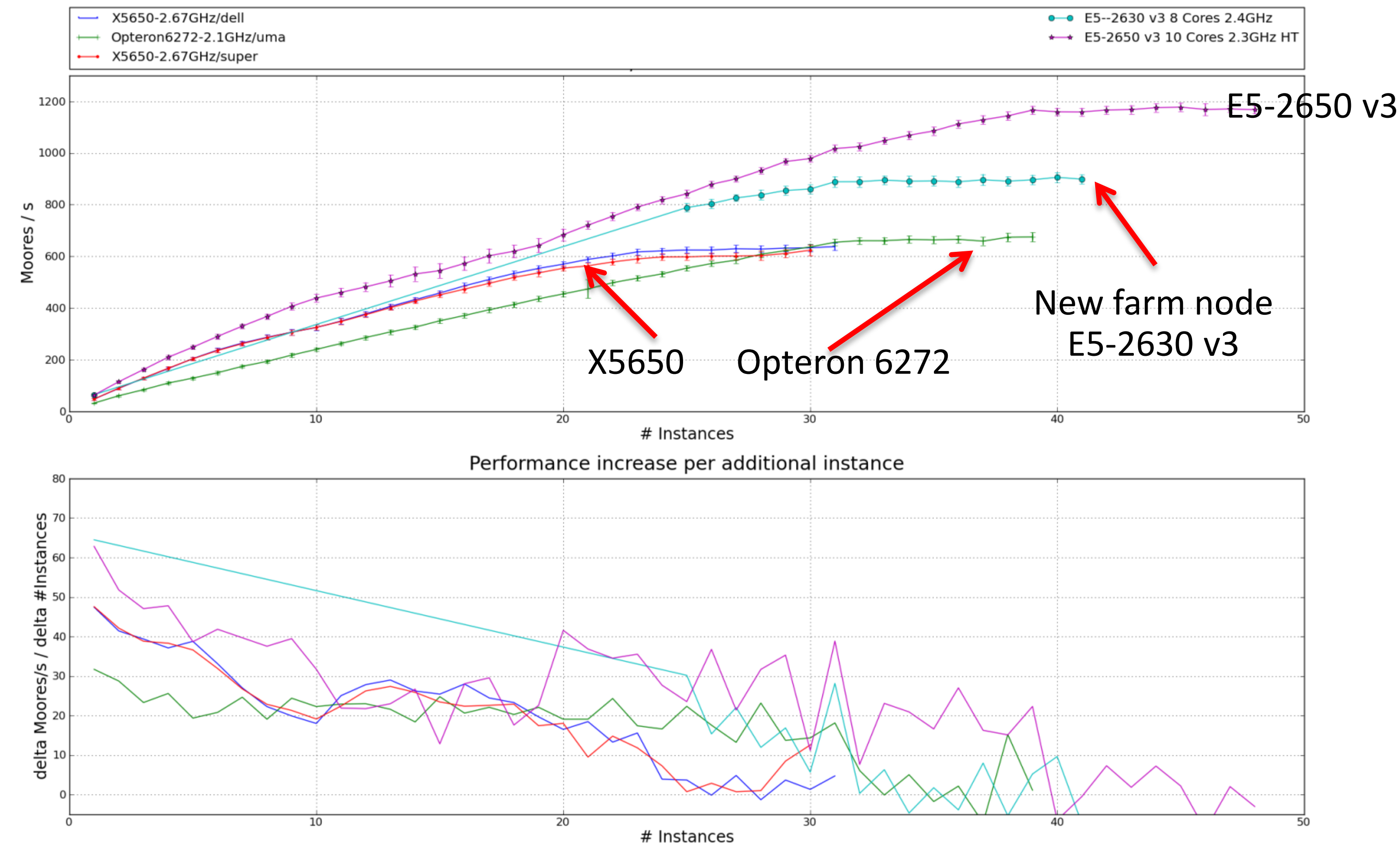
# Application Profile

- Processes very short lived, highly independent tasks, a.k.a Events
  - 200-300 ms per Event
- No inherent multi-threading
  - Parallelism is achieved by launching multiple instances
  - Launch N instances of program
  - Choose N for max machine throughput
- Mixture of strongly branching code and floating point operations
- Memory footprint is approximately 1.1 GB
  - 600 MB static
  - 400-500 MB dynamic
- Processes are created by fork()ing a master process
  - Reduces memory footprint
  - Accelerates startup
  - Has some issues → TBD



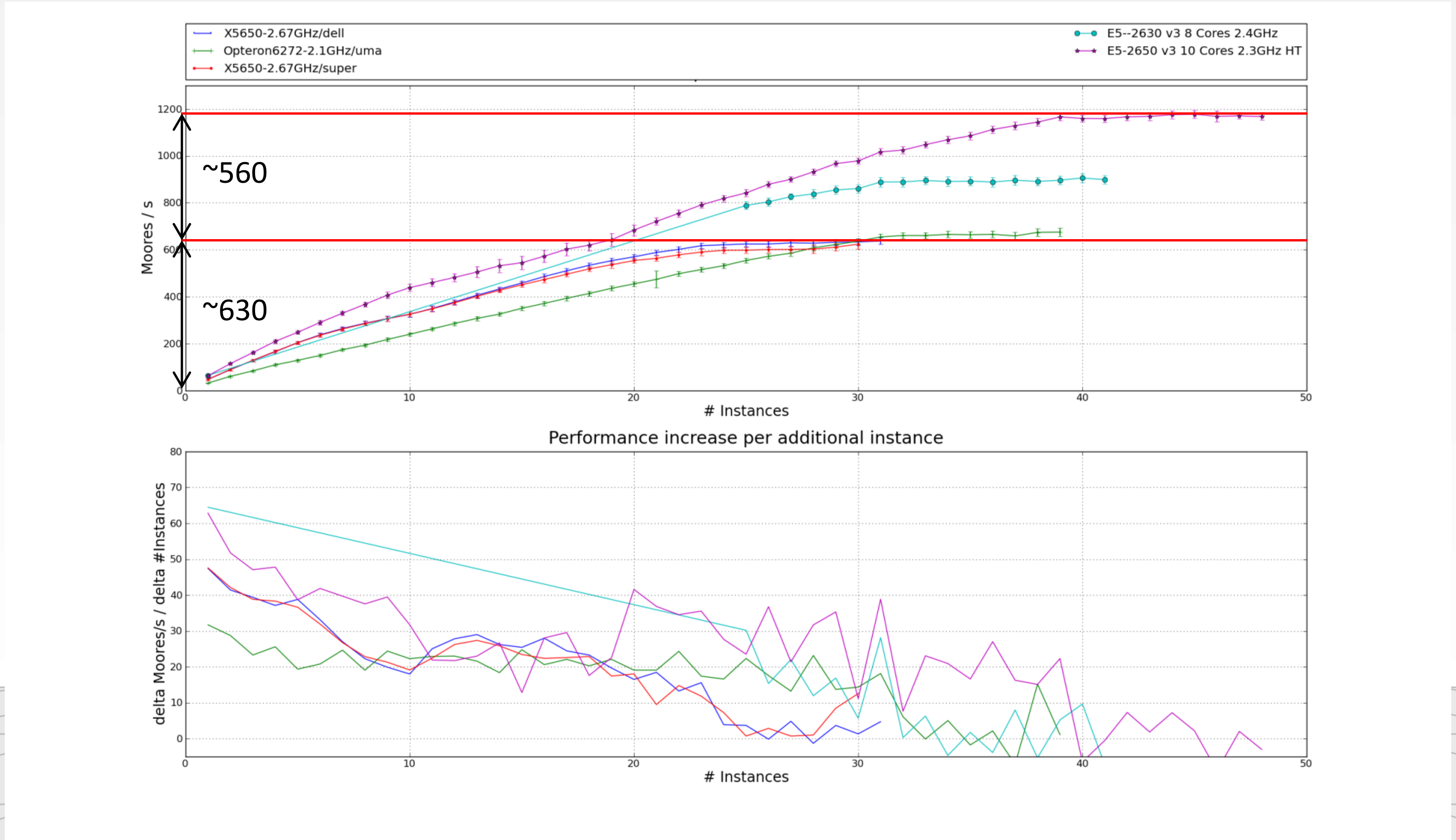
# Results

# Benchmark Output




- Plot 1: Machine throughput vs. number of running instances
- Plot 2: Increase in throughput per additional instance
- Benchmark scans the optimum number of program instances
- Results from previous generation and Haswell Dual Socket farm nodes

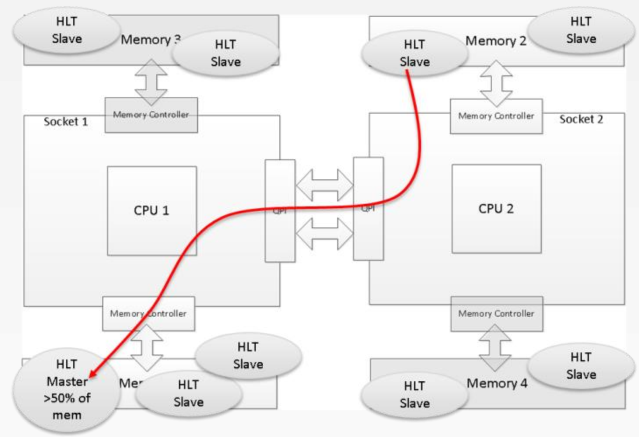
# Interesting little detail



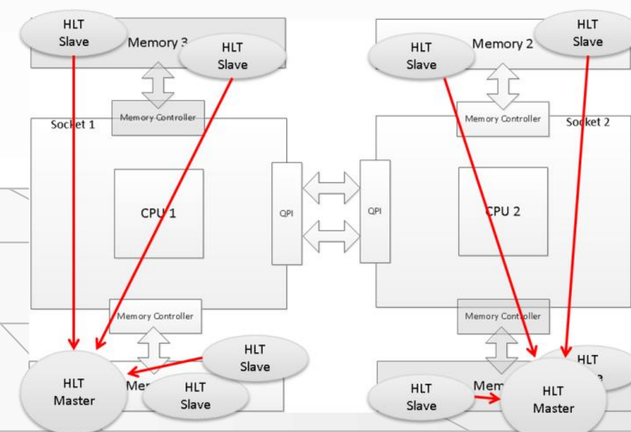

- Program seems to run faster on first socket than on second
  - Effect is also visible on old generation nodes, but to lesser extent

# Consequences of NUMA Architecture

- Fork() master process is scheduled to a particular NUMA node and allocates most of its memory there
- Children are distributed over the 2 sockets depending on number of children
  - First to the socket with master
  - Then other socket(s)
- Off master processes are hit by additional latency in QPI/HT hop
- Spawn as many masters as sockets
  - Lock master and children to specific NUMA node with numactl
  - Up to 14% performance gain in machine throughout!
- Haswell 10+ core CPUs are internally divided into two NUMA nodes
  - 2-3% additional gain
  - Enable 'Cluster On Die' (COD) feature in BIOS

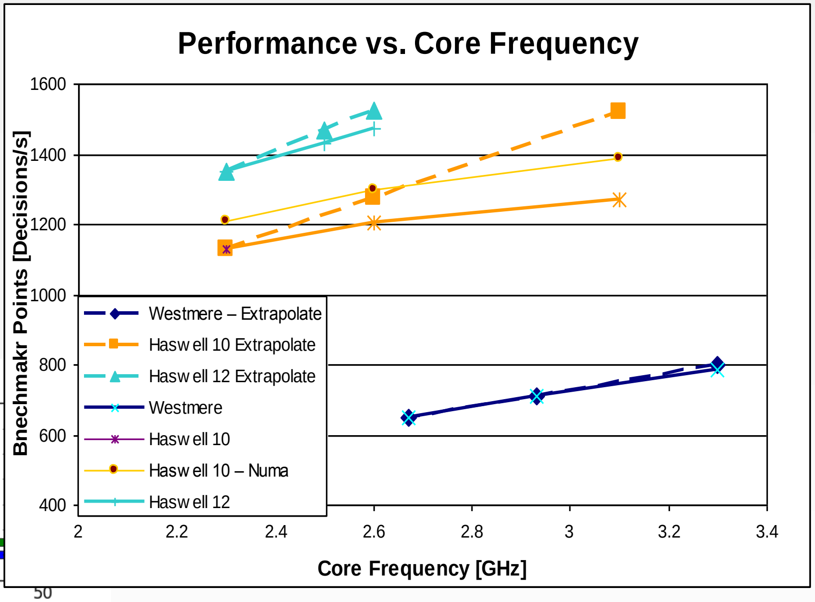


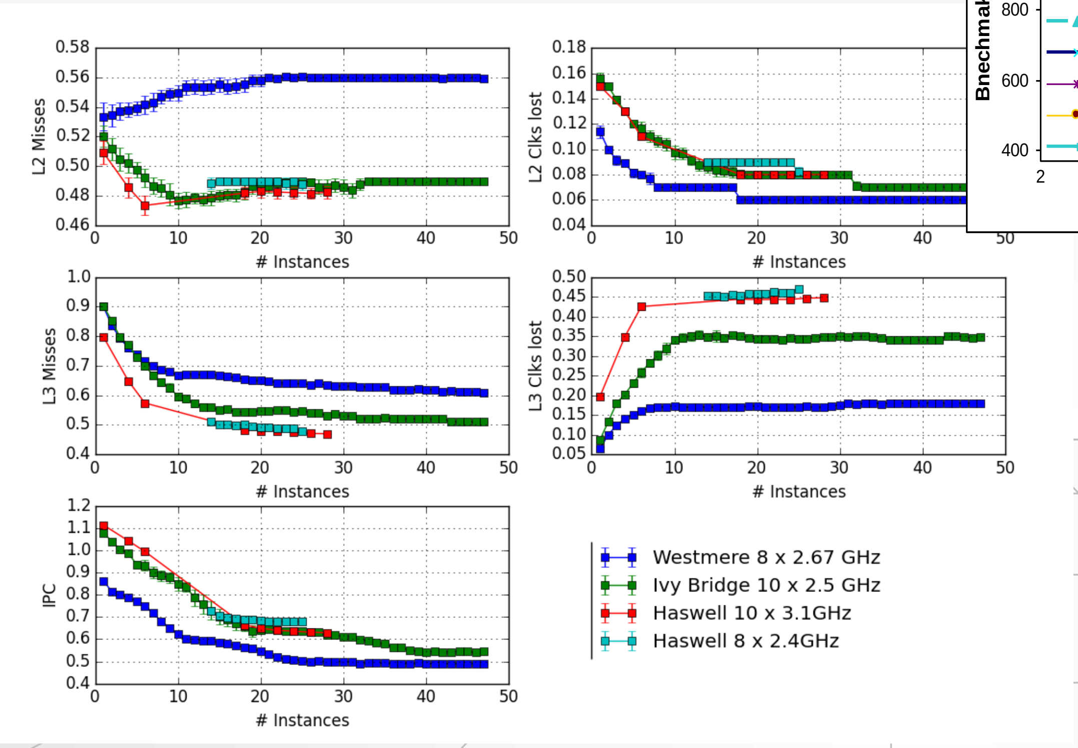

| CPU | Decisions/s No NUMA | Decisions/s NUMA | NUMA Gain |
|---|---|---|---|
| Intel X5650 (8 cores) | 599.6 | 648.8 | 1.08 |
| Opteron X272 | 632.35 | 682 | 1.08 |
| E_2630 v3 (8 cores) | 865 | 986 | 1.14 |
| E_2650 v3 (10 cores) | 1129 | 1210 | 1.07 |



# Other Memory Issues

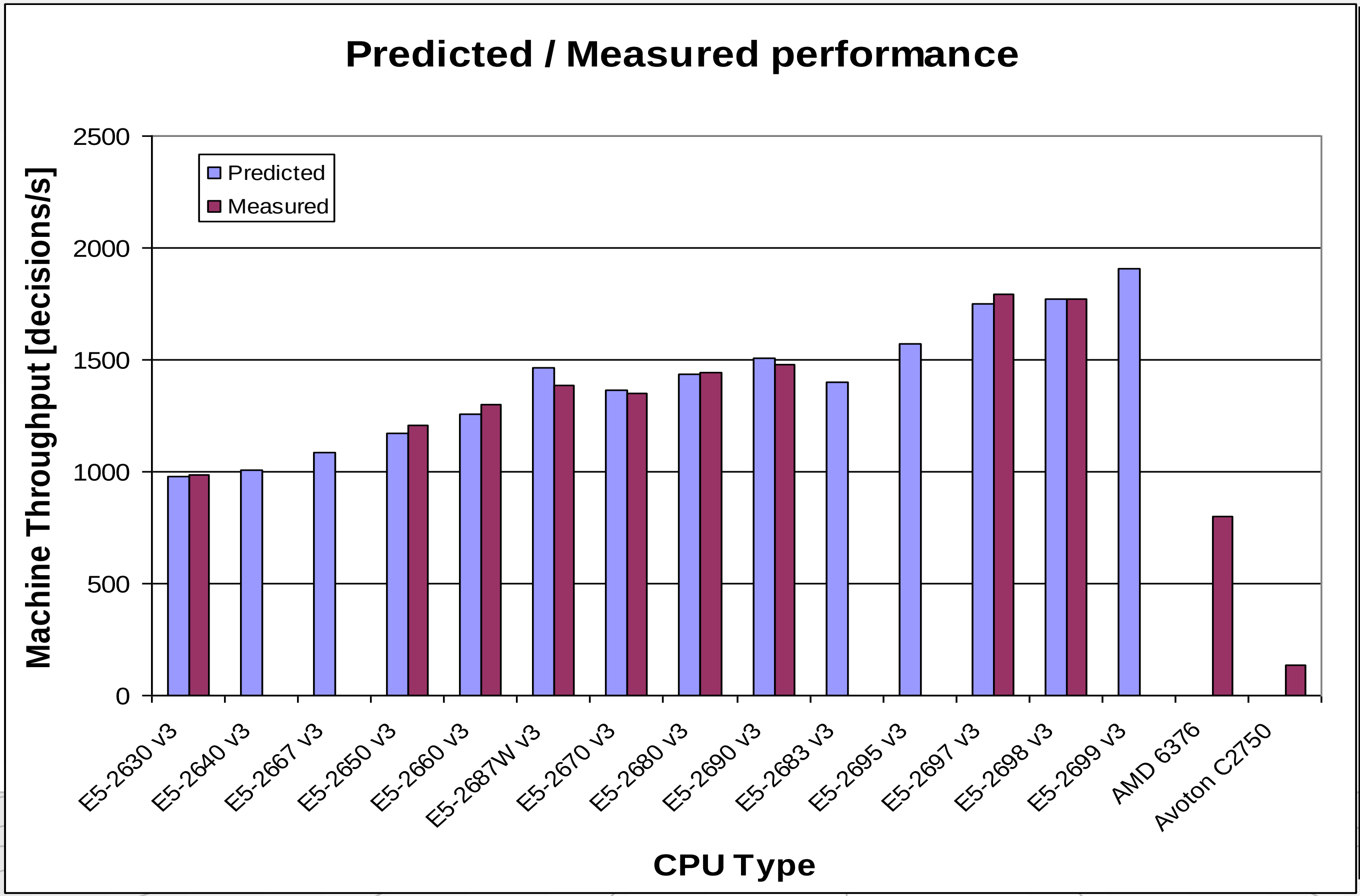
- Memory Latency seems to finally have caught up with our application
- Westmere: 2 MB/Core
- Ivy: 2.5 MB/Core
- Haswell 2.5 MB/Core
- Tool: Intel Performance Counter Monitor




- While L3 Miss rate has gone down
- Each L3 Miss loses more clock cycles now
- DDR4 memory has better throughput but currently higher latency!
- Potential for more optimization
- Should become better in the future
  - DDR4 is quite new

# Performance Overview

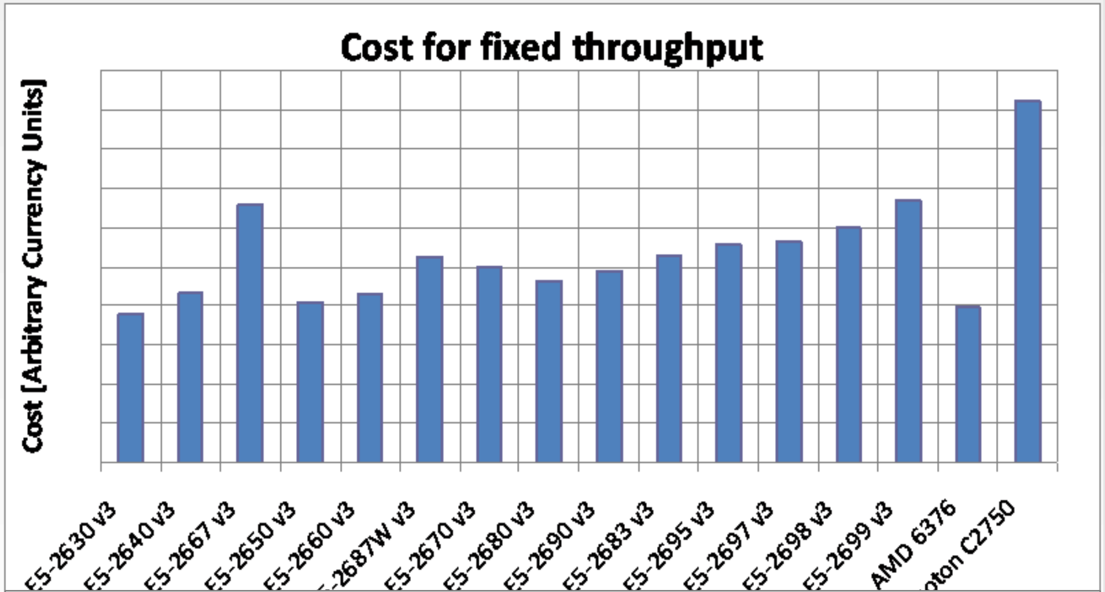


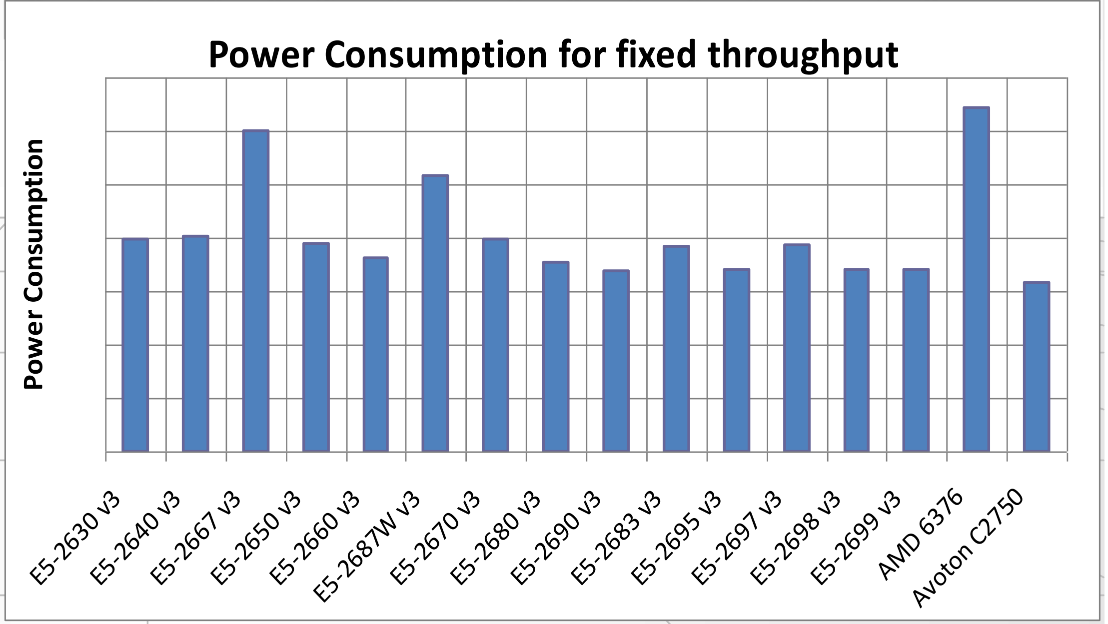

| CPU | Cores | Freq |
|---|---|---|
| 2630 v3 | 8 | 2.4 |
| 2640 v3 | 8 | 2.6 |
| 2667 v3 | 8 | 3.2 |
| 2650 v3 | 10 | 2.3 |
| 2660 v3 | 10 | 2.6 |
| 2687W v3 | 10 | 3.2 |
| 2670 v3 | 12 | 2.3 |
| 2680 v3 | 12 | 2.5 |
| 2690 v3 | 12 | 2.6 |
| 2683 v3 | 14 | 2.0 |
| 2695 v3 | 14 | 2.3 |
| 2697 v3 | 14 | 2.6 |
| 2698 v3 | 16 | 2.3 |
| 2699 v3 | 18 | 2.3 |
| AMD 6373 | 16 | 2.3 |
| C2750 | 8 | 2.4 |

- Performance model based on benchmark and memory measurements
- Assumes a dual socket system

# Costs / Power

- Cost of dual socket system for fixed throughput
  - 24 blades for Avoton
- Costs included:
  - Main board
  - Memory
  - PSU
  - CPU
  - Chassis
  - Double memory required for 2687W and above
  - AMD6376 and Avoton based on system quotes
- Power consumption
- Measured
  - Avoton
  - AMD 6376
  - E5-2630 v3
  - E5-2690 v3
- Other Power consumption estimated
  - 1.6 x TDP x $N_{sockets}$
  - Fits all measured systems within 5%



# A word about AMD and Avoton

- AMD
  - Pretty good value for price (not much choice)
  - Power consumption is an issue
- Avoton
  - Actually Better than portrayed here
  - Suffers from two key shortcomings in our case
    - No Hyperthreading
    - Very little cache per core (512 k)
    - → Instructions per Clock cycle: < 0.4
    - If your code is optimized for this it should be quite efficient
  - Many slow systems
    - Network overhead
    - Administration overhead
    - HDDs (if you need them)

# Conclusion

- We have created a stand alone version of our High Level Trigger
- Significantly improved Trigger performance with NUMA awareness
- Memory access optimization has become even more critical with latest Intel CPU generation
  - Both Xeon and Avoton
- Selected E5-2630 v3 as most cost efficient platform for our new farm
- This does not mean it's also the best platform for you though

# Acknowledgements


- Companies / Institutions helping with access to test platforms and power measurements
- CERN IT
- E4 Computer Engineering
- Intel Swindon Labs
- ASUS
- Macle GMBH

# Questions