TECHNISCHE UNIVERSITÄT DRESDEN

Michael Kluge, ZIH

Correlating Multiple TB of Performance Data to User Jobs

Lustre User Group 2015, Denver, Colorado

Zellescher Weg 12
Willers-Bau A 208
Tel. +49 351 - 463 – 34217

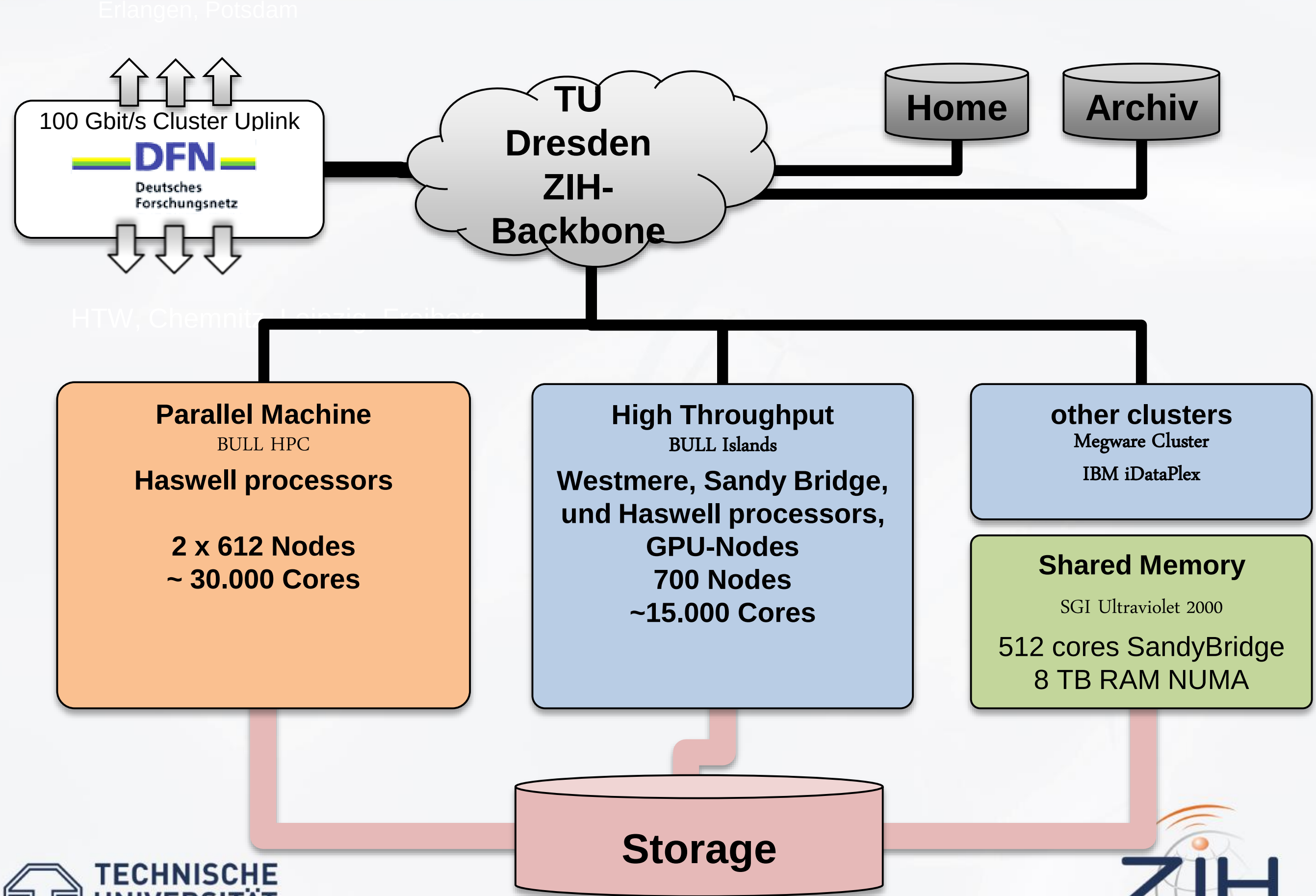
Michael Kluge (michael.kluge@tu-dresden.de)

ZIH
Center for Information Services & High Performance Computing

About us

- ZIH: about us
  - HPC and service provider
  - Research Institute
  - Big Data Competence Center
- Main research areas:
  - Performance Analysis Tools
  - Energy Efficiency
  - Computer Architecture
  - Standardization Efforts (OpenACC, OpenMP)

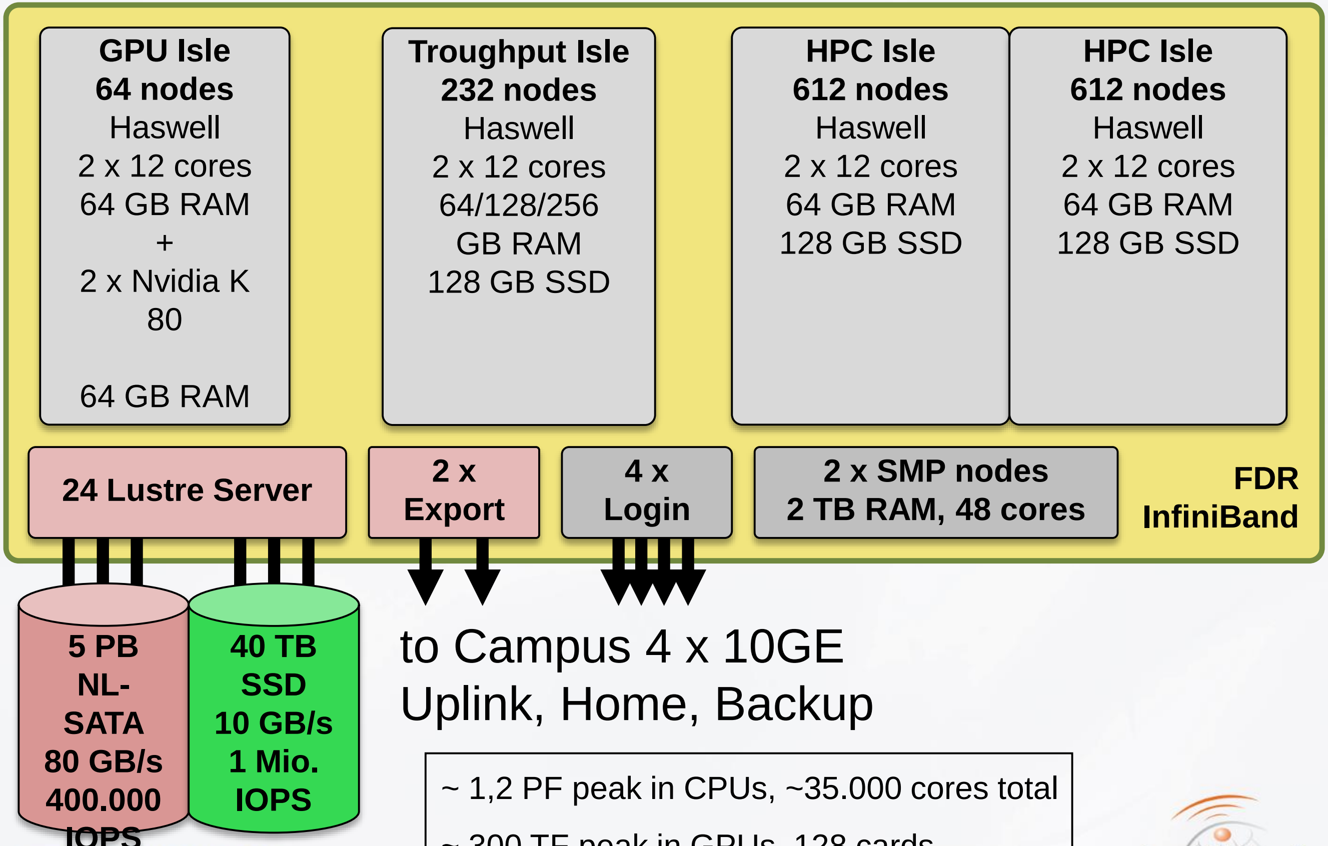
ZIH HPC and BigData Infrastructure



TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

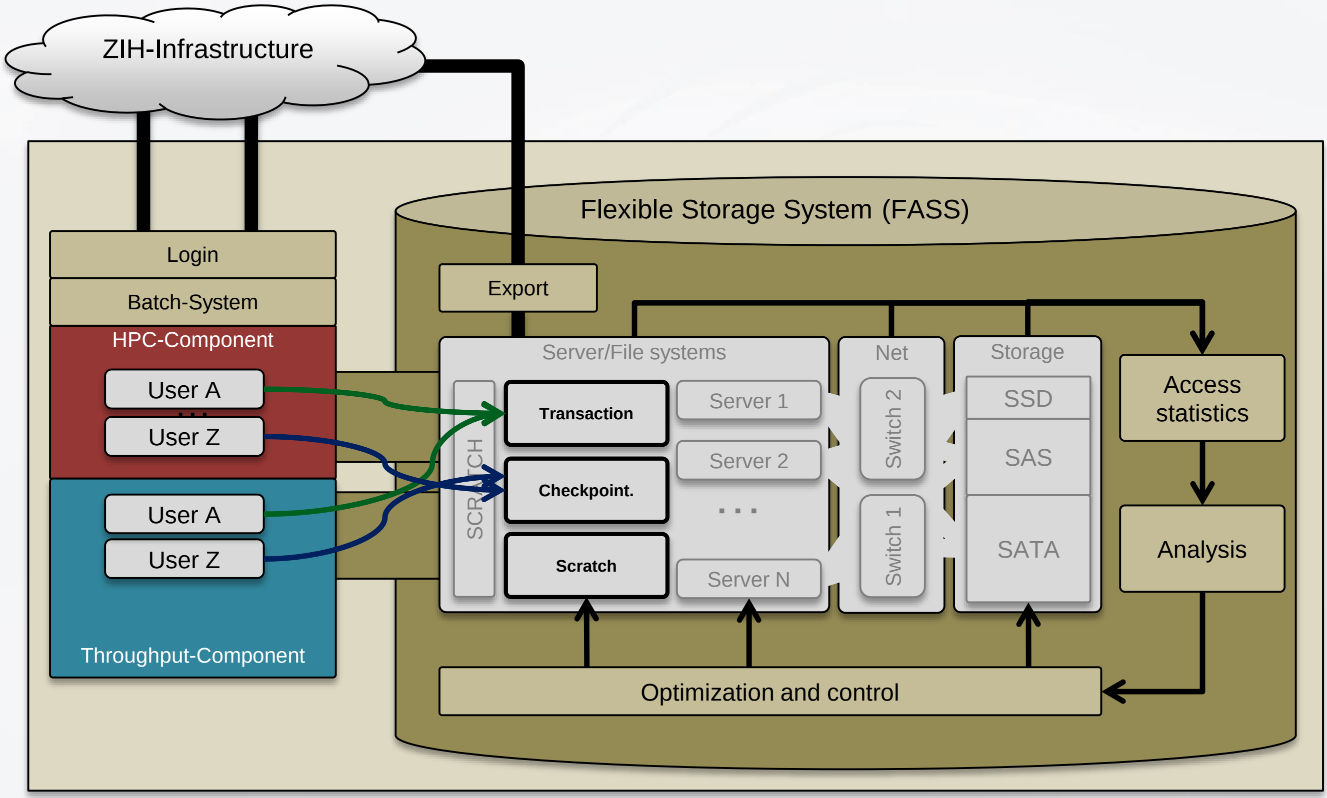
HRSK-II, Phase 2, currently installed



~ 1,2 PF peak in CPUs, ~35.000 cores total

~ 300 TF peak in GPUs, 128 cards

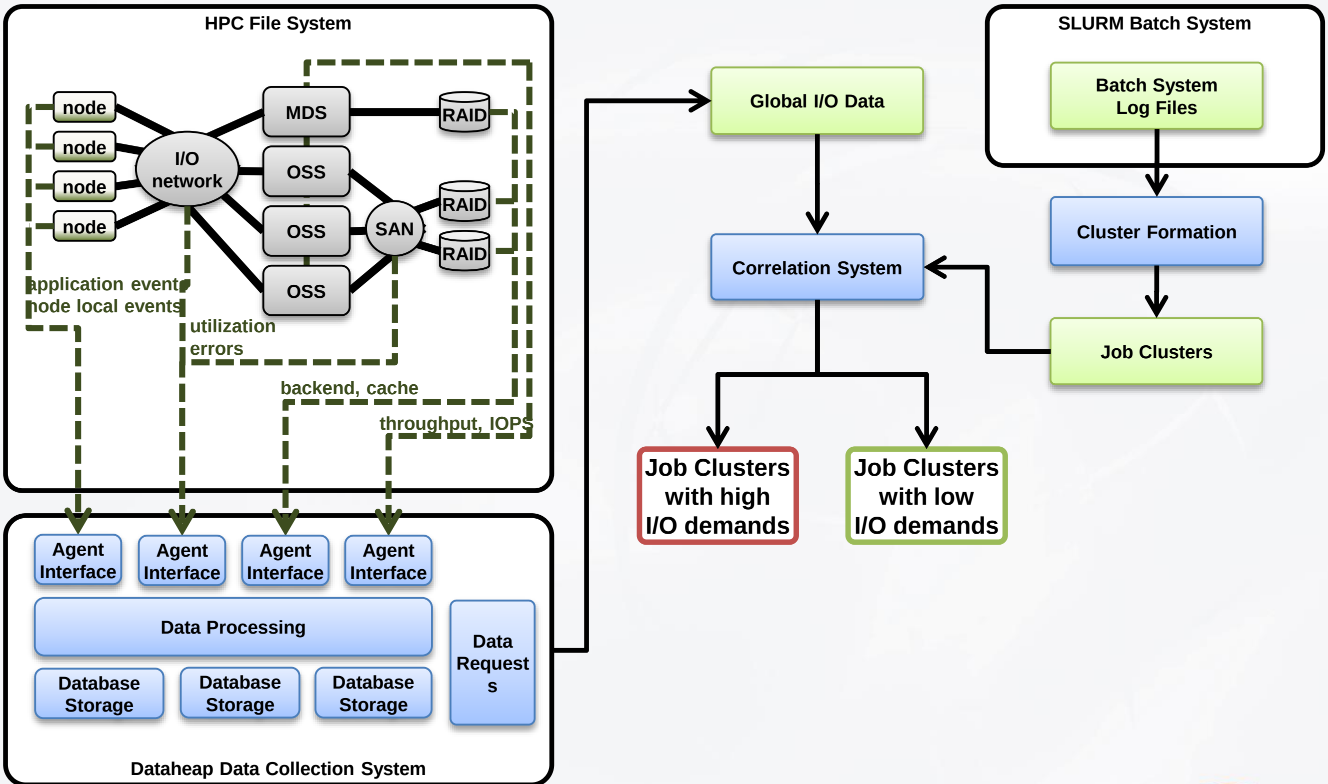
Architecture of our storage concept (FASS)



Performance Analysis for I/O Activities (1)

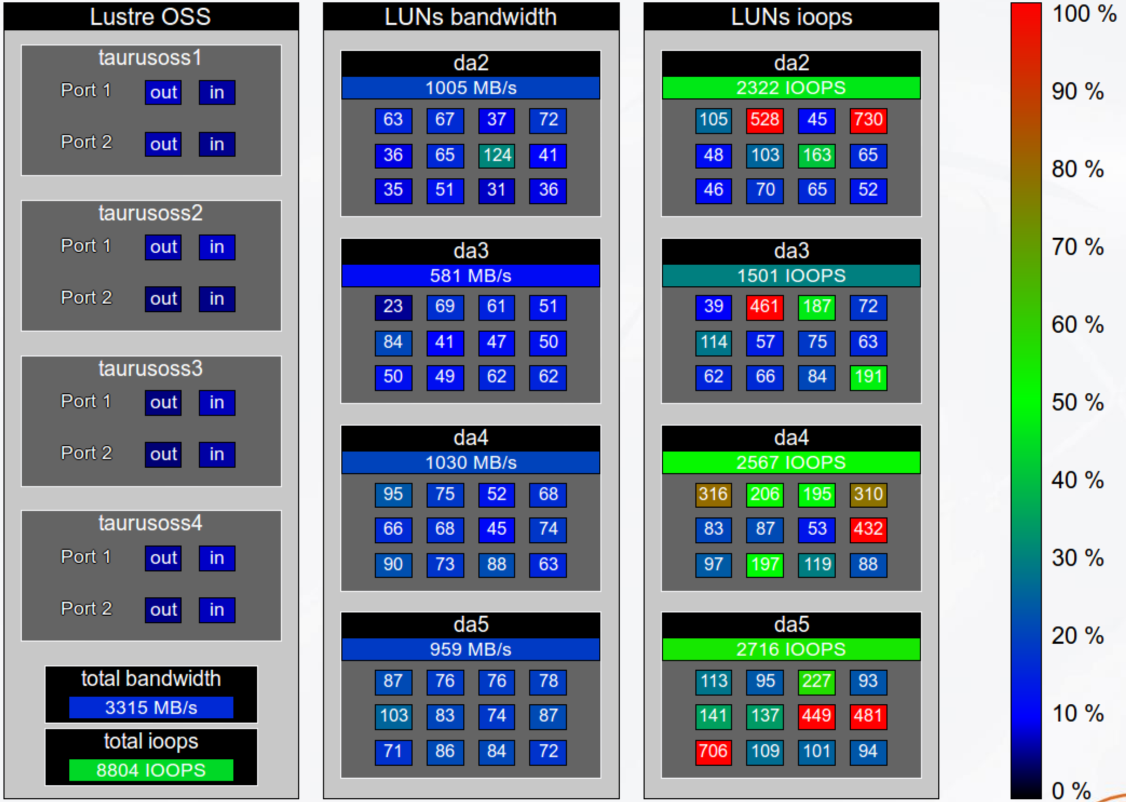
- What is of interest
  - Application requests, Interface types
  - Access sizes/patterns
  - Network/Server/Storage utilization
- Level of detail:
  - Record everything
  - Record data every 5 to 10 seconds
- Challenge:
  - How to analyze this?
  - How to deal with anonymous data?

I/O and Job Correlation: Architecture/Software

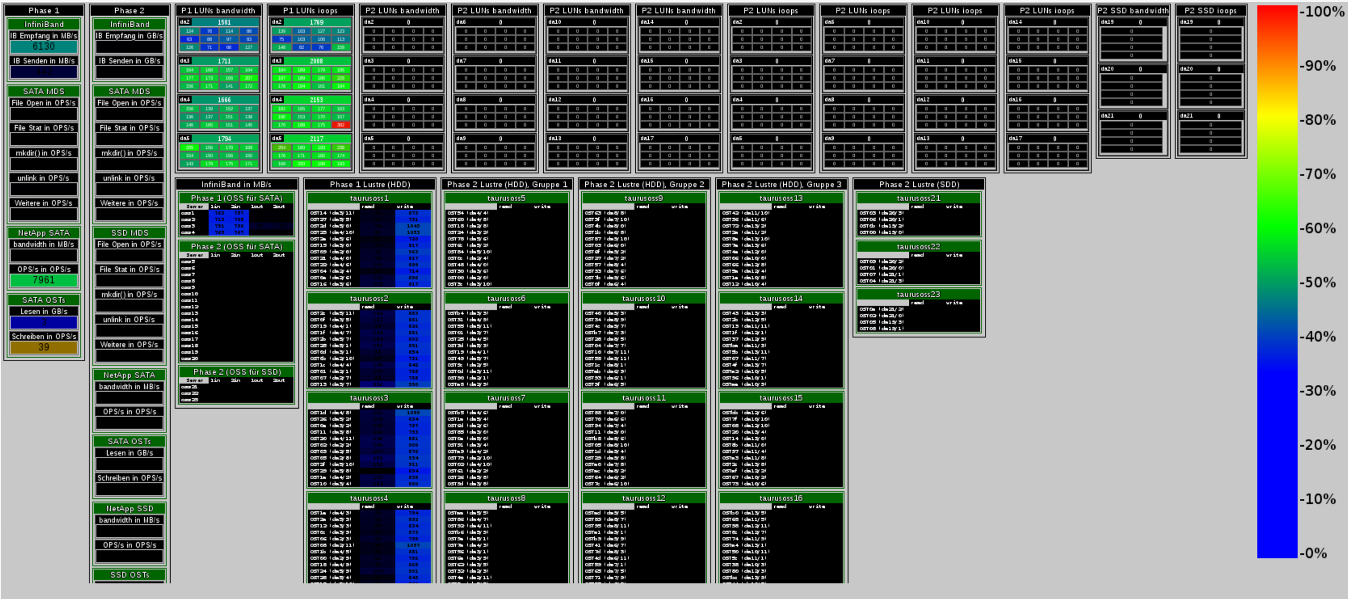


https://fusionforge.zih.tu-dresden.de/projects/dataheap/

Visualization of Live Data for Phase 1



Visualization of Live Data for Phase 2



Performance Analysis for I/O Activities (2)

- Amount of collected data:
  - 240 OSTs, 20 OSS servers, …
  - Looking at stats and brw_stats (how to store a histogram in database?)
  - → ~75.000 tables
  - about 1 GB of data per hour
- Analysis is supposed to cover 6 month
  - 4 TB data
  - No way to analyze this with any serial approach

Analysis Process

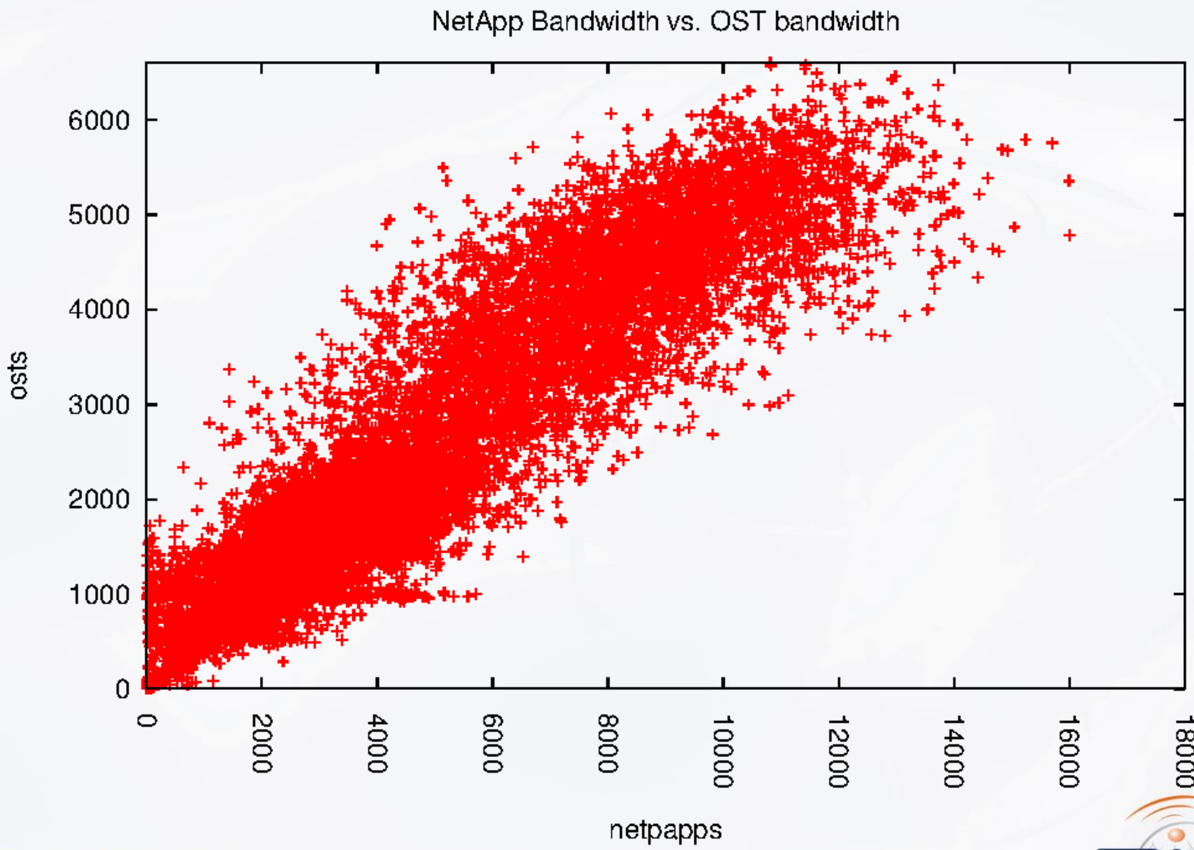
- map operations to a set of tables
  - modify all tables (align data, scale, aggregate over time)
  - create a new table from an input set of tables (aggregation)
  - plot (time series, histograms)
- currently: single thread Haskell program
- Swift/T work in progress
  - 4 TB means 50 seconds on an 80 GB/s file system for the initial read
  - Swift/T has nice ways to describe data dependencies and to parallelize this kind of workflows (open for suggestions!)

Approach to look at the Raw Data

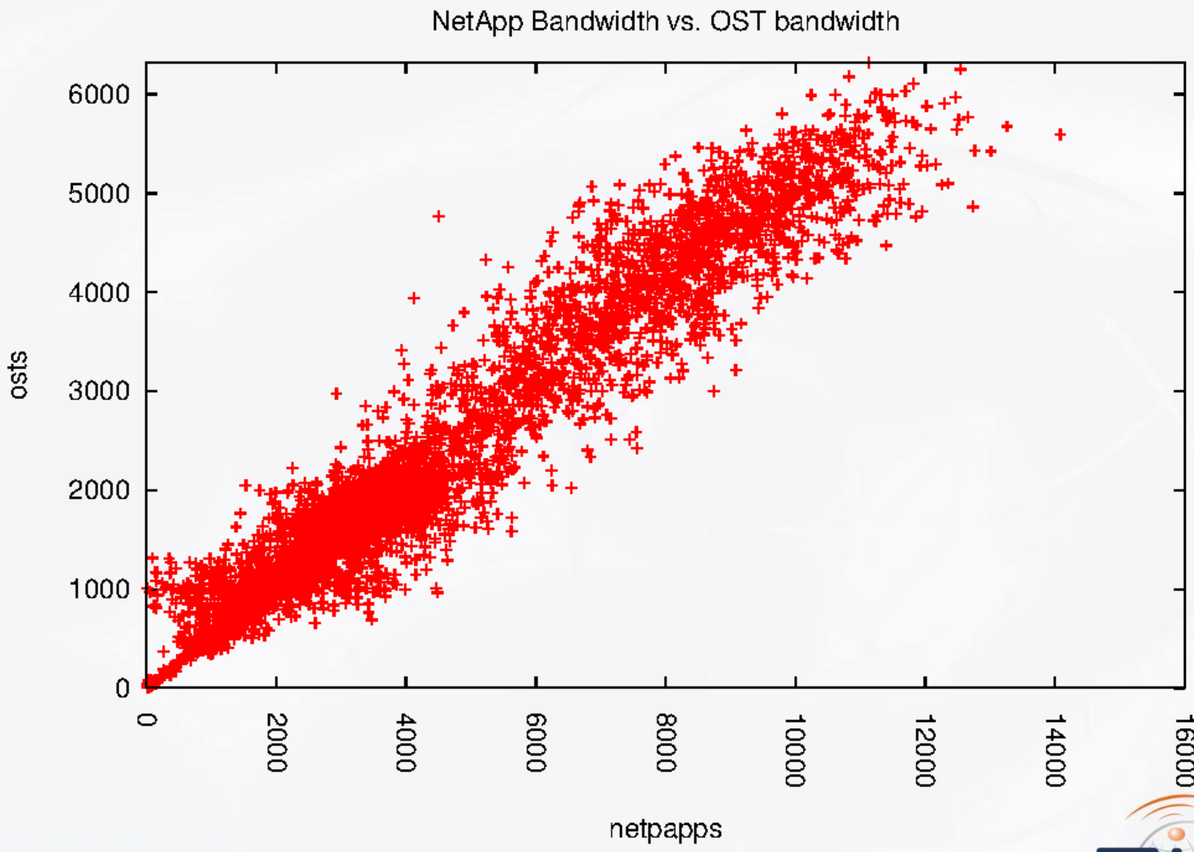
- analysis of the raw performance data
  - take original data and create derived metrics
  - look at plots and histograms

- correlate metrics with user jobs
  - take job log and split jobs into classes
  - check for correlations between job classes and performance metrics

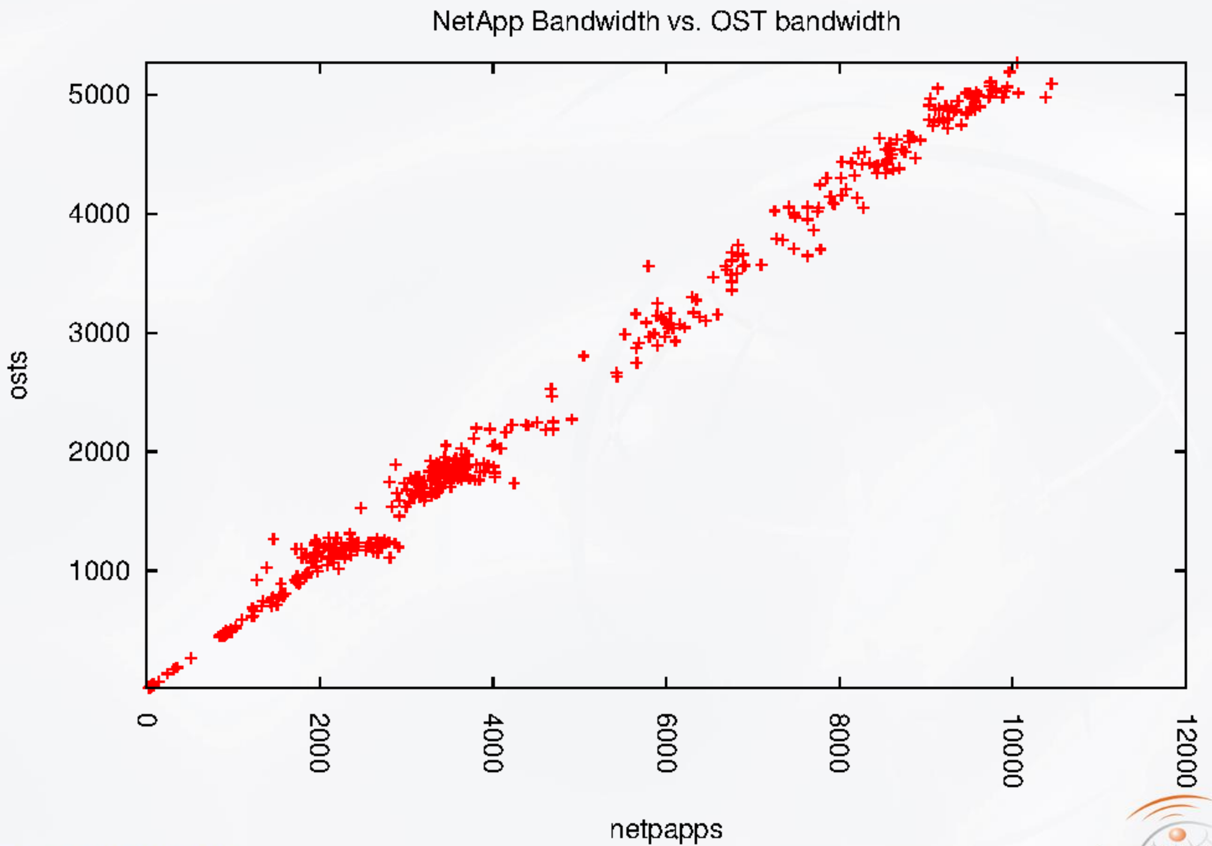
Metric Example: NetApp vs. OST Bandwidth (20s)



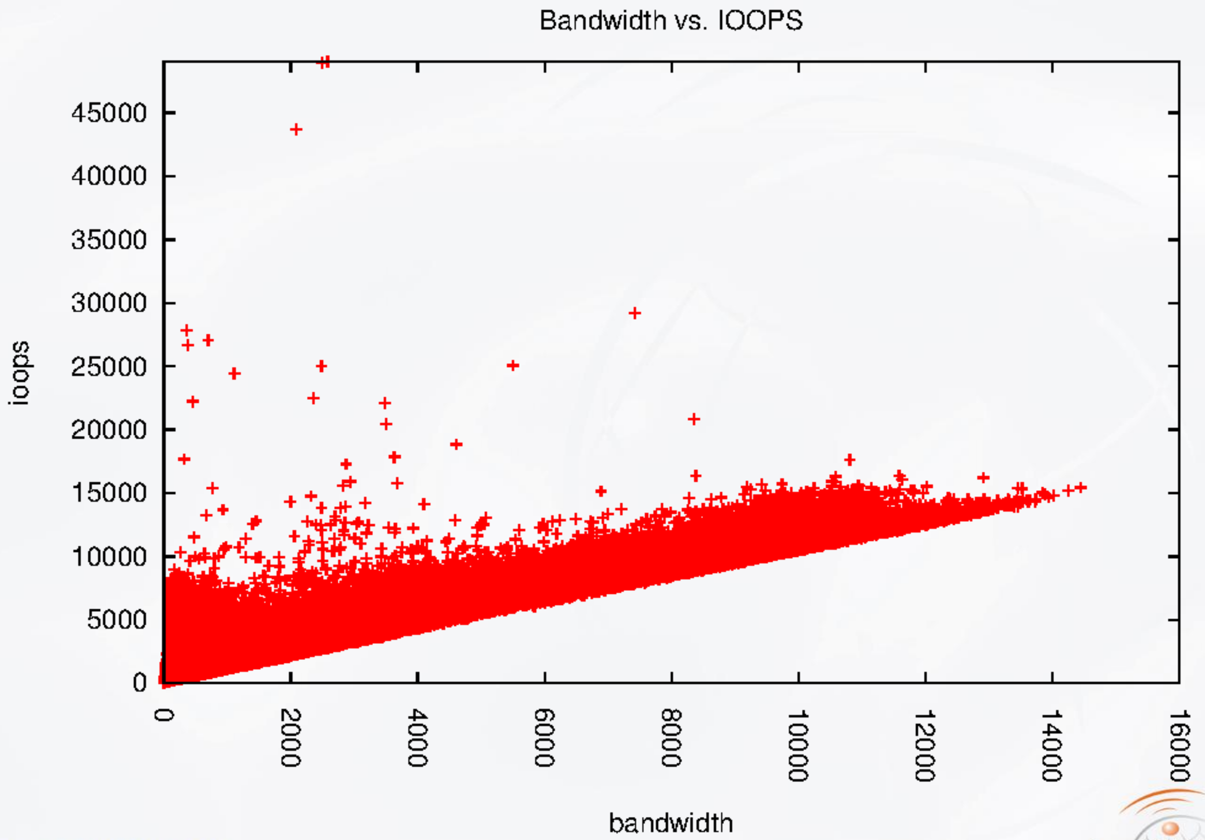
Metric Example: NetApp vs. OST Bandwidth (60s)



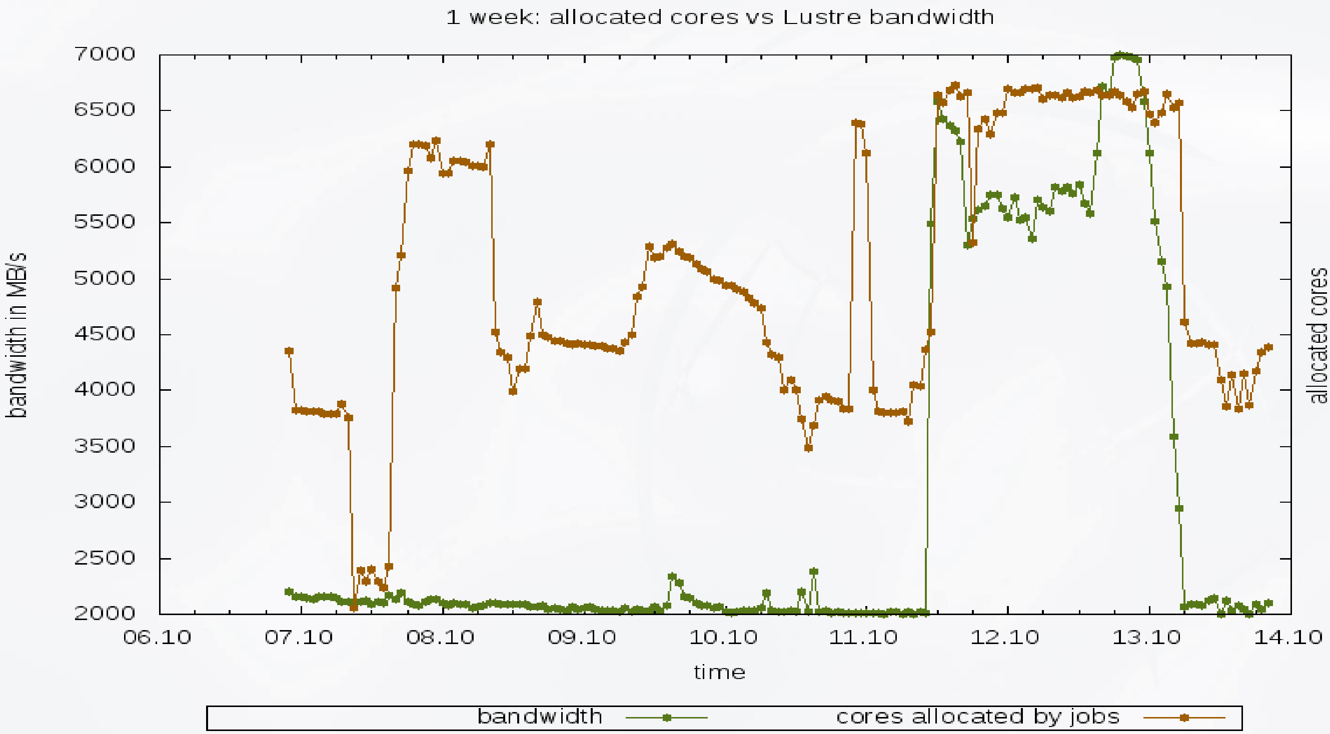
Metric Example: NetApp vs. OST Bandwidth (10 minutes)



Metric Example: 3 Month bandwidth vs. IOPS



I/O and Job Correlation: Starting Point (Bandwidth)



Building Job Classes

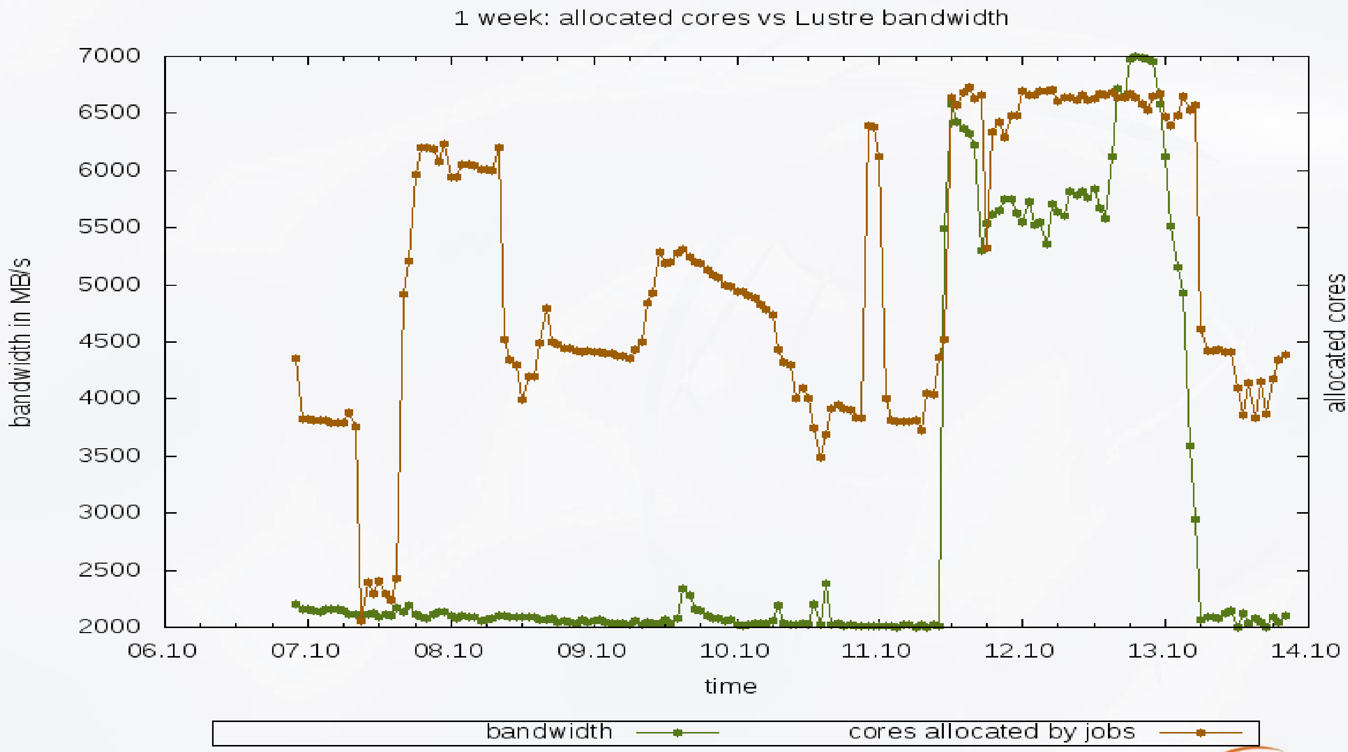
- a class is defined by:
  - user id, core count
  - similar runtime
  - similar start time (time isolation)

Require: $objs$ // set of objects

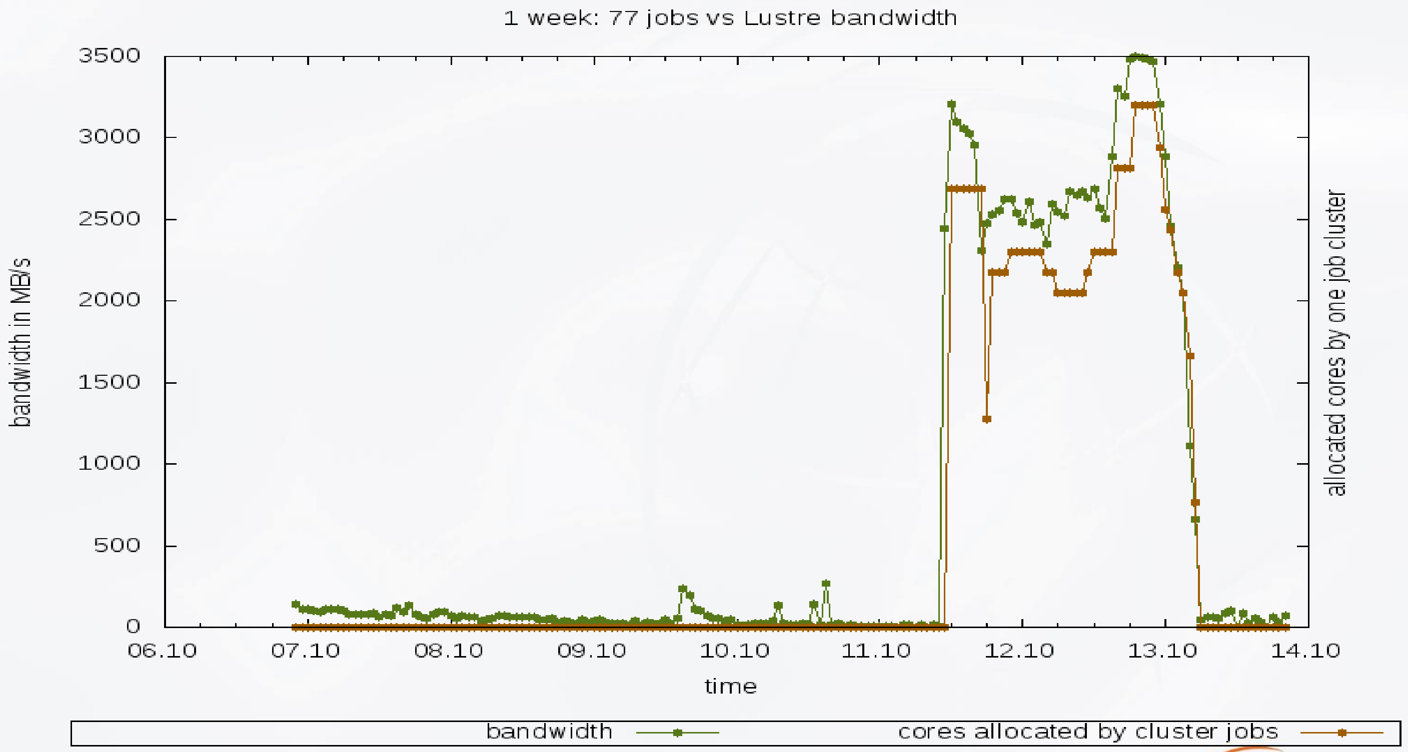
$num = 0$
sort $objs$ ordered by $linearorder$
insert $objs[0]$ into $clusters[num]$
for $i = 1; i < |objs|; i++$ **do**
    if $d(objs[i-1], objs[i]) > d_{max}$ **then**
        $num++$
    end if
    insert $objs[i]$ into $clusters[num]$
end for

- sort all jobs by runtime, split list at gaps
- split these groups again if they are not overlapping (long gap between start)
- challenges: runtime fluctuation, large run time variations

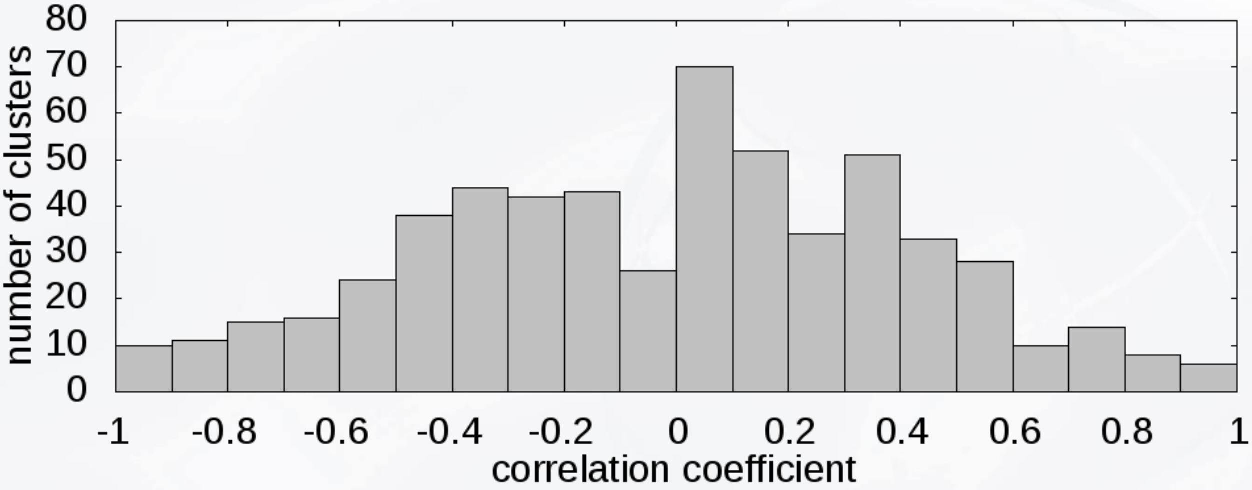
I/O and Job Correlation: Preliminary Result (1)



I/O and Job Correlation: Preliminary Result (1)



I/O and Job Correlation: Correlation Factors



Open Topics

- How to store a histogram in database?
- Other approaches than Swift/T
- Deal with the zeros
- Cache intermediate results

Conclusion / Questions

- it is possible to map anonymous performance data back to user jobs
- advance towards storage controller analysis (caches, backend bandwidth, …)