Leibniz Supercomputing Centre

Continuous System and Application Monitoring

Dr. Matthias Brehm
brehm@lrz.de

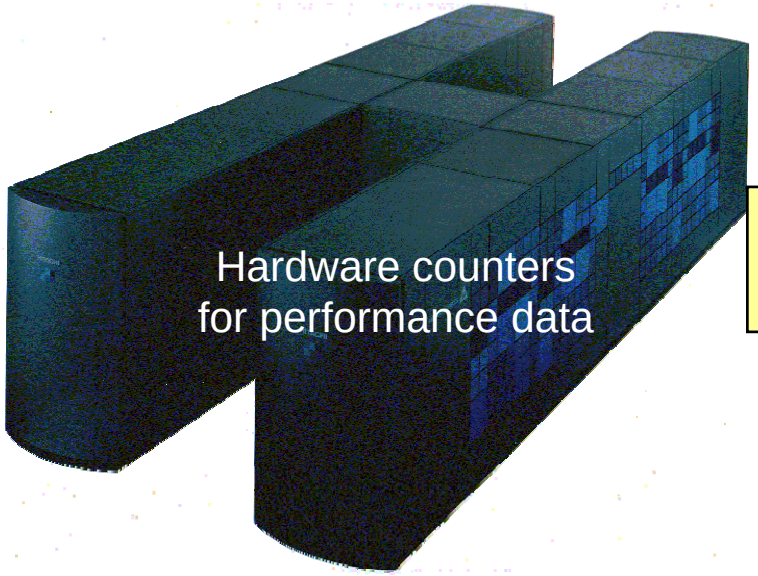
October 08

Continous Monitoring

- **Use of continous monitoring**
  - **State and health of system**
  - **User behaviour**
  - **Application performance**

- **Ensure quality**
  - **from the system operations point of view**
    - automatic detection of bottlenecks in the system or configuration
    - detection of malfunctions, e.g. in the interconnect network or in the batch queueing system
  - **wrt the job mix:**
    - How can optimal usage of resources be achieved
    - Use of historic data to predict optimal job mix or configuraton
  - **wrt to a single application**
    - to automatically detect bottlenecks in applications
    - detect Deltas
      - *Optimization*
      - *Modifications*
      - *Topologiy*
      - *Input data sets*

Early approaches at LRZ



cron job
SAR
(every 5 minutes)

Text files with snapshots of performance and resouce usage

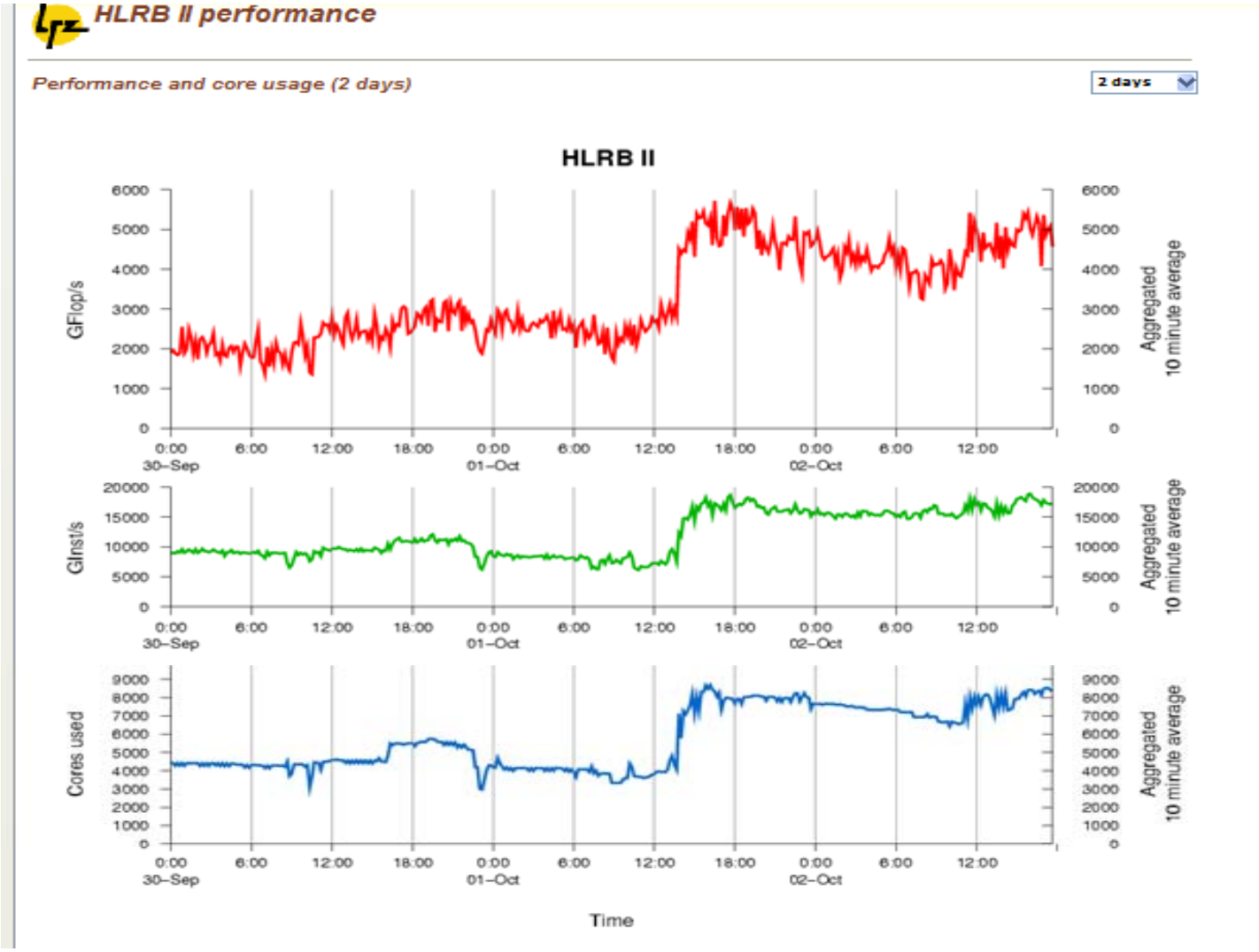
command line interface
SARBEN
for grouping, sorting, and filtering

Text files with performance data grouped by jobs, users, and projects

SARBEN GUI
for grouping, sorting, filtering, judging, joining project info...

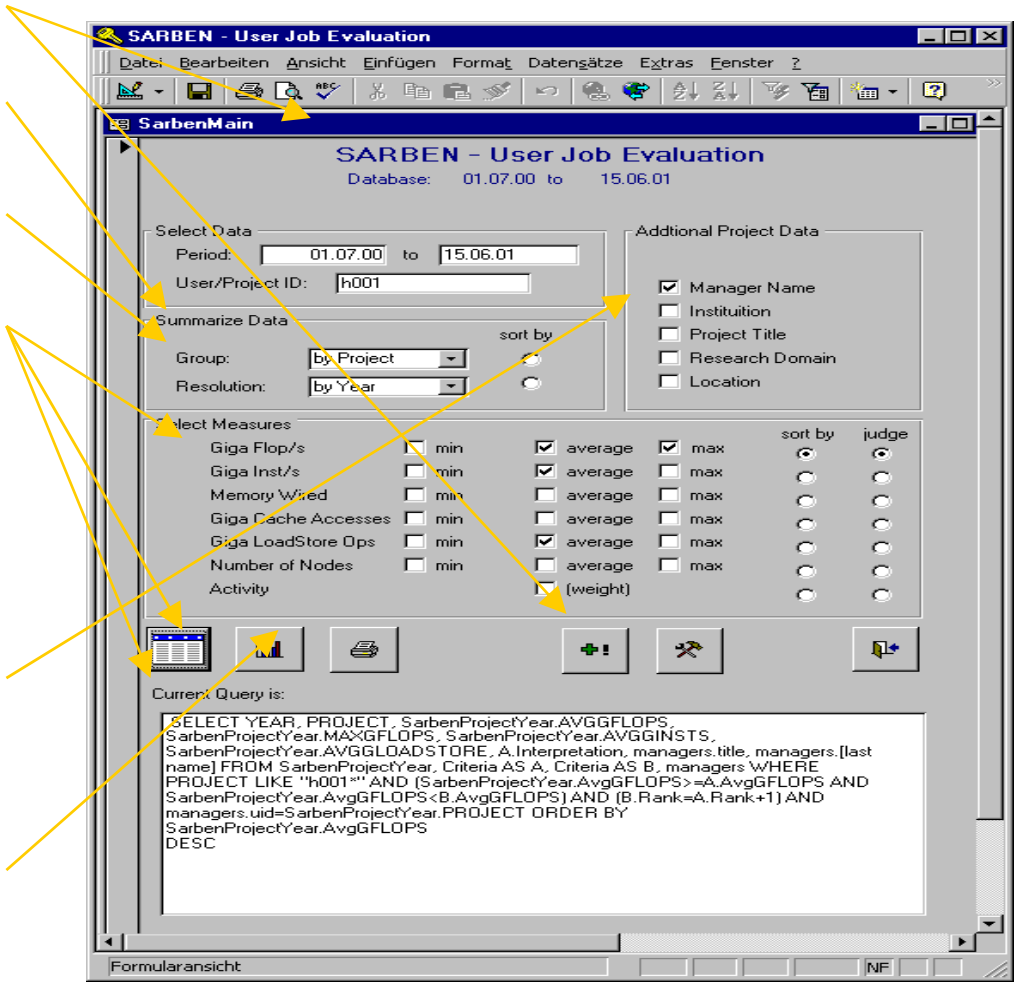
DB tables

Health of System
Continous Performance Monitoring

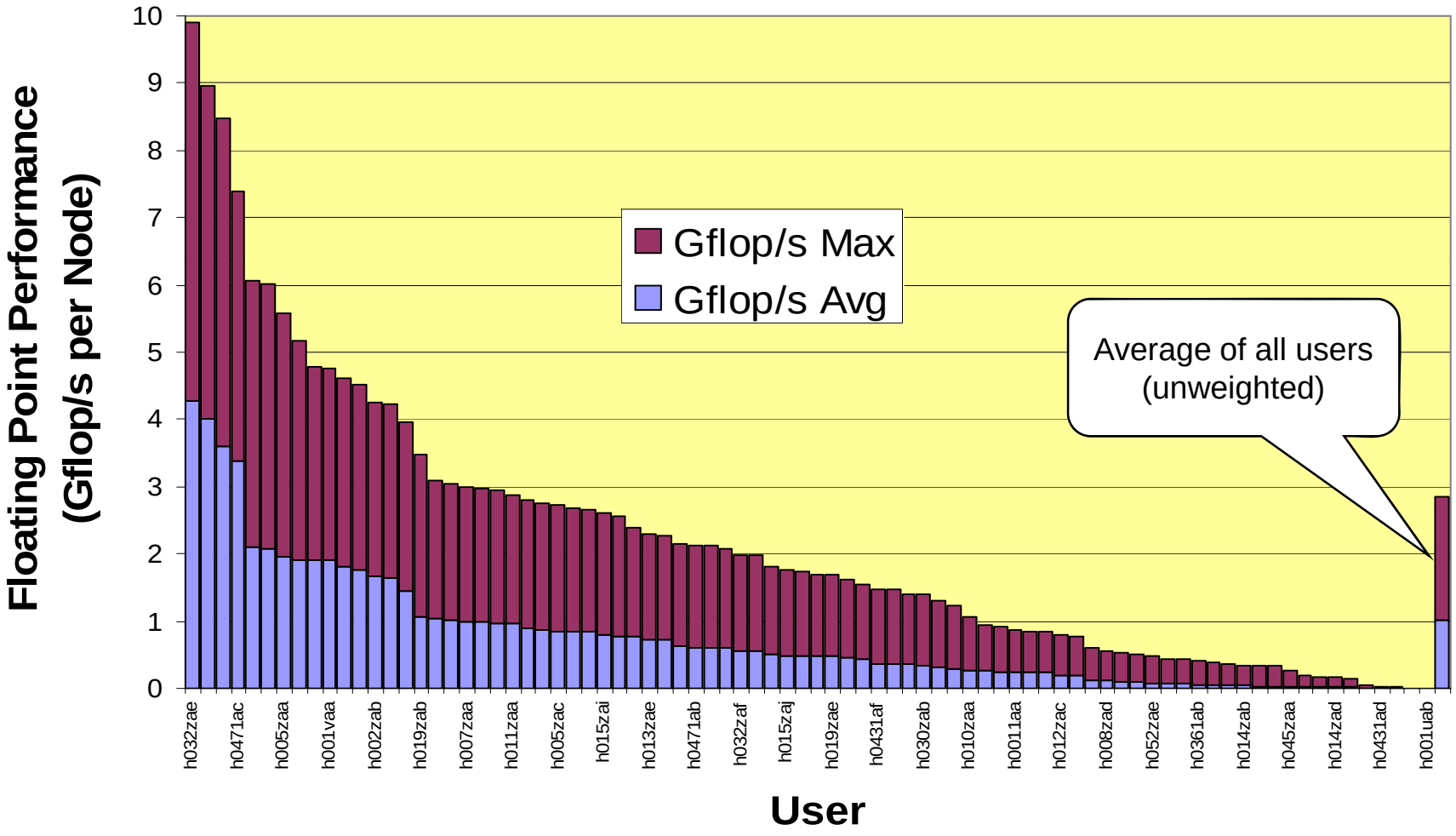


Early approaches at LRZ
SARBEN GUI: Graphical User Interface

- Job based records can be merged with Database.
- Tables on per-user, per-project, per-day, per-month, and per-year period are generated in advance.
- SQL queries on those tables can be generated using the form (GUI).
- Additional features via SQL:
  - sorting and filtering by any of the database/query columns
  - judgement on performance relative to predefined short criteria table
  - join with data of project proposals for easier lookup and manager contact
  - further grouping values: e.g. Flops/Insts, Cache/Mem
- chart generation



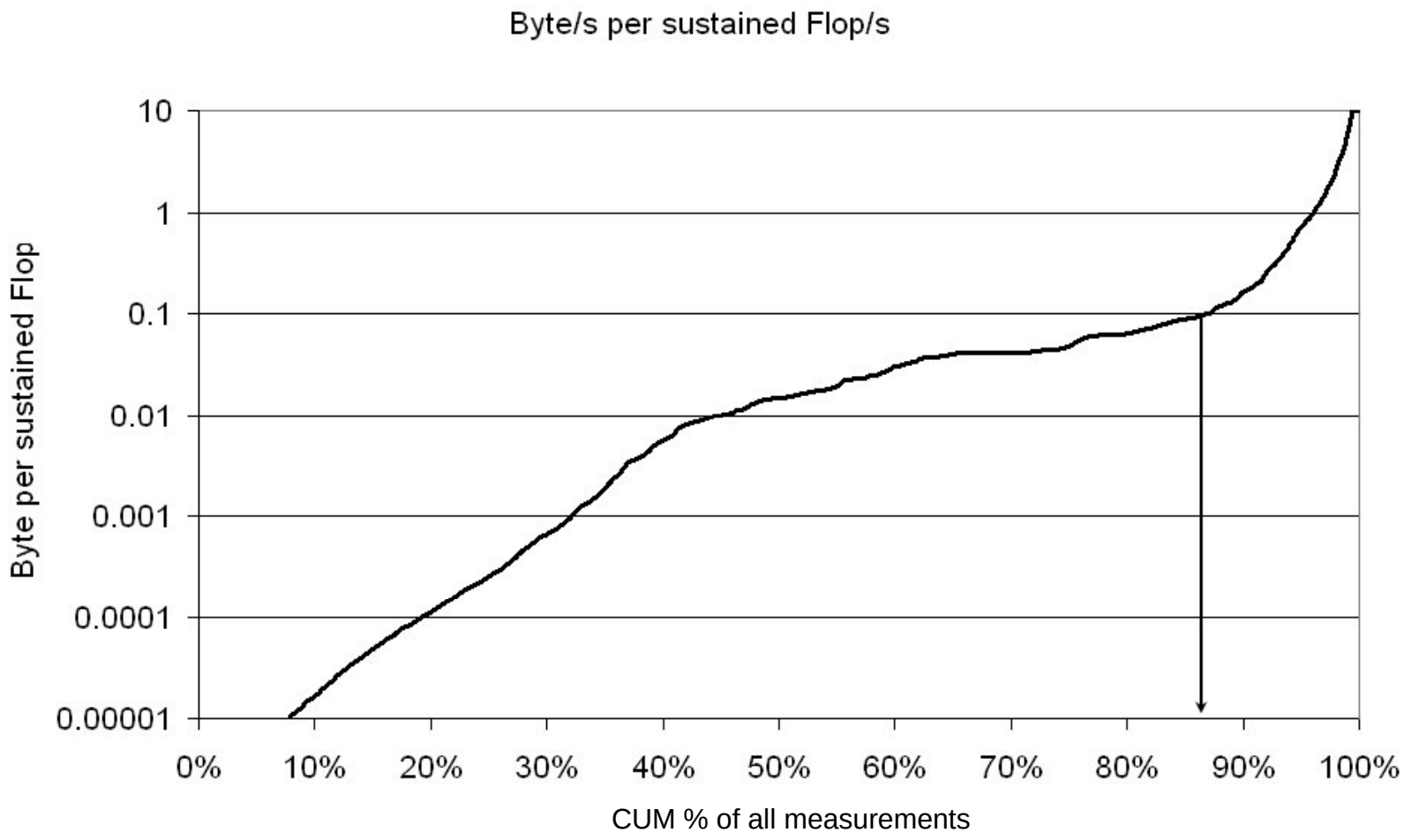
Ranking of User Performance (based on 5 minutes Intervals)



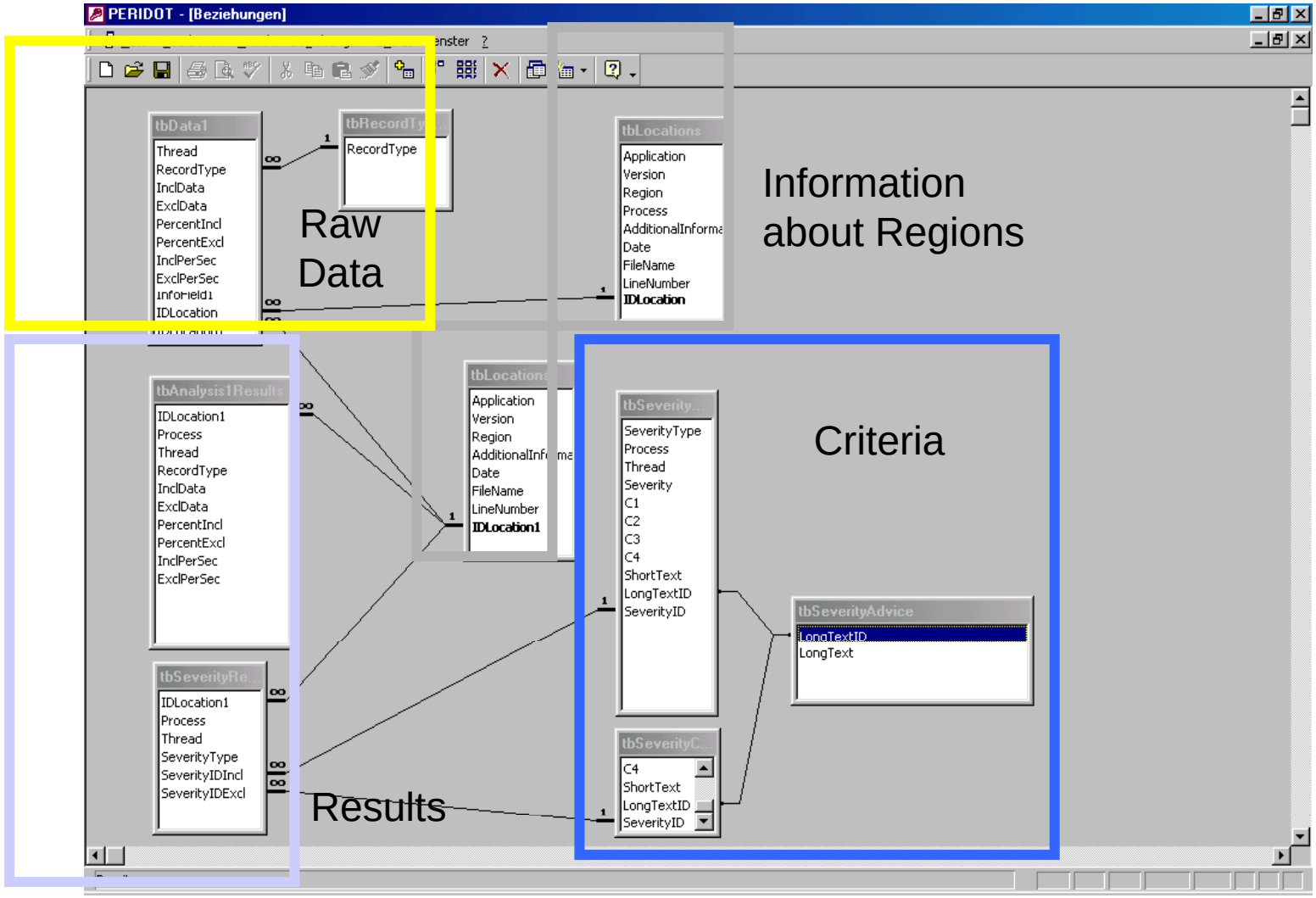
Percentage of Peak Performance at LRZ „Sustained Performance“

System	Year	Avg. % of FP Peak Performance
CRAY X-MP/Y-MP	1989-1995	33 %
CRAY T90	1996-2001	25 %
Fujitsu VPP	1997-2004	20 %
IBM SP2	1995-2001	11 %
Hitachi SR8000	2000-2006	12.5 %
Itanium Linux Cluster	2003-	10 %
SGI Altix	2006-	9 %

Bandwith requriements of all applications based on monitored data of SR8000



LRZ PERIDOT Database Structure
Instrumented Regions



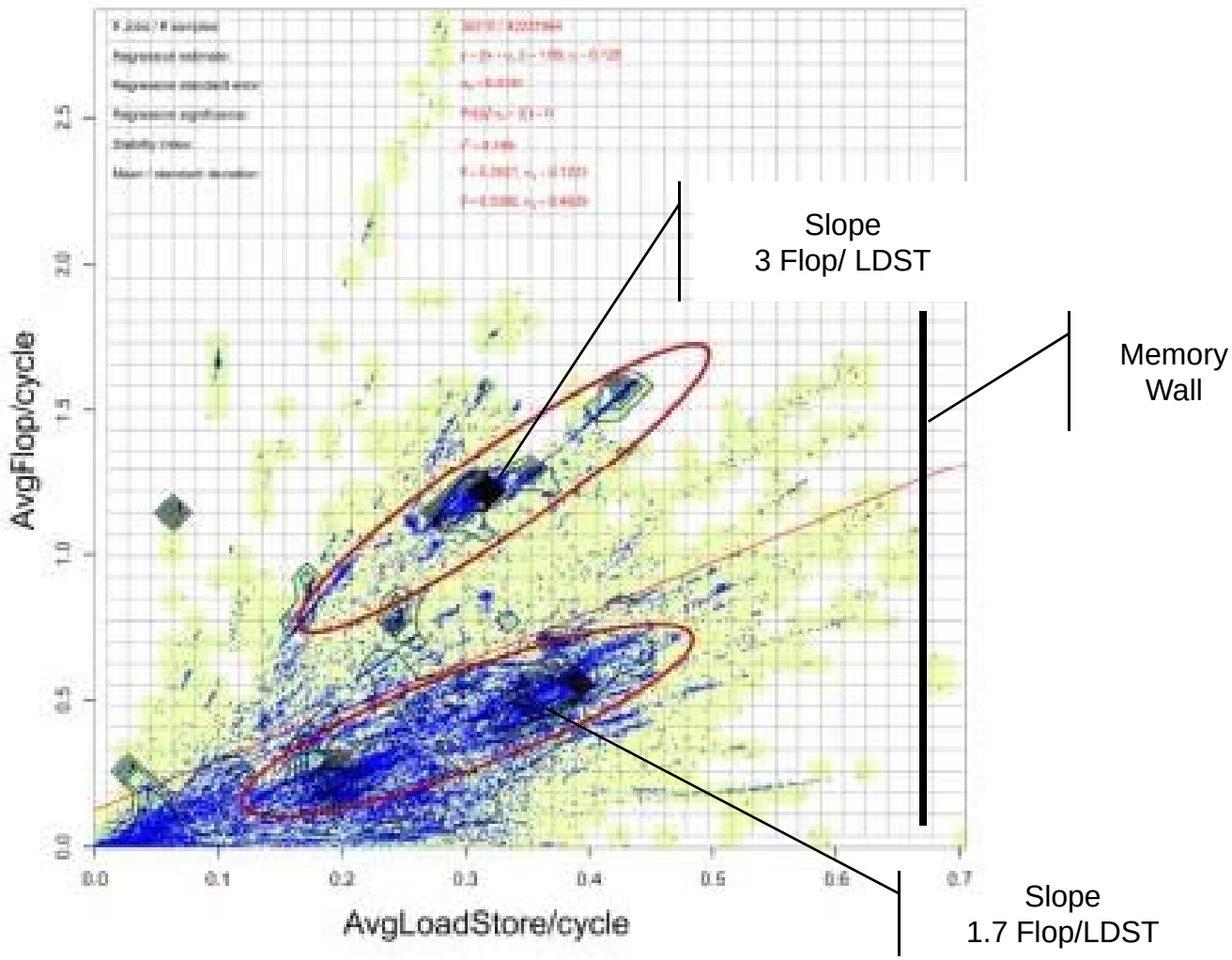
Criteria

- **Severity:** CPU time is significantly different from elapsed time
- **Criteria:** CPU/Elased Time **<**7.00000E-01
- **Description:**
- This region shows a CPU time which is substantially less than the elapsed time.
- **Advice:**
- Look for system calls, I/O, memory allocation etc.

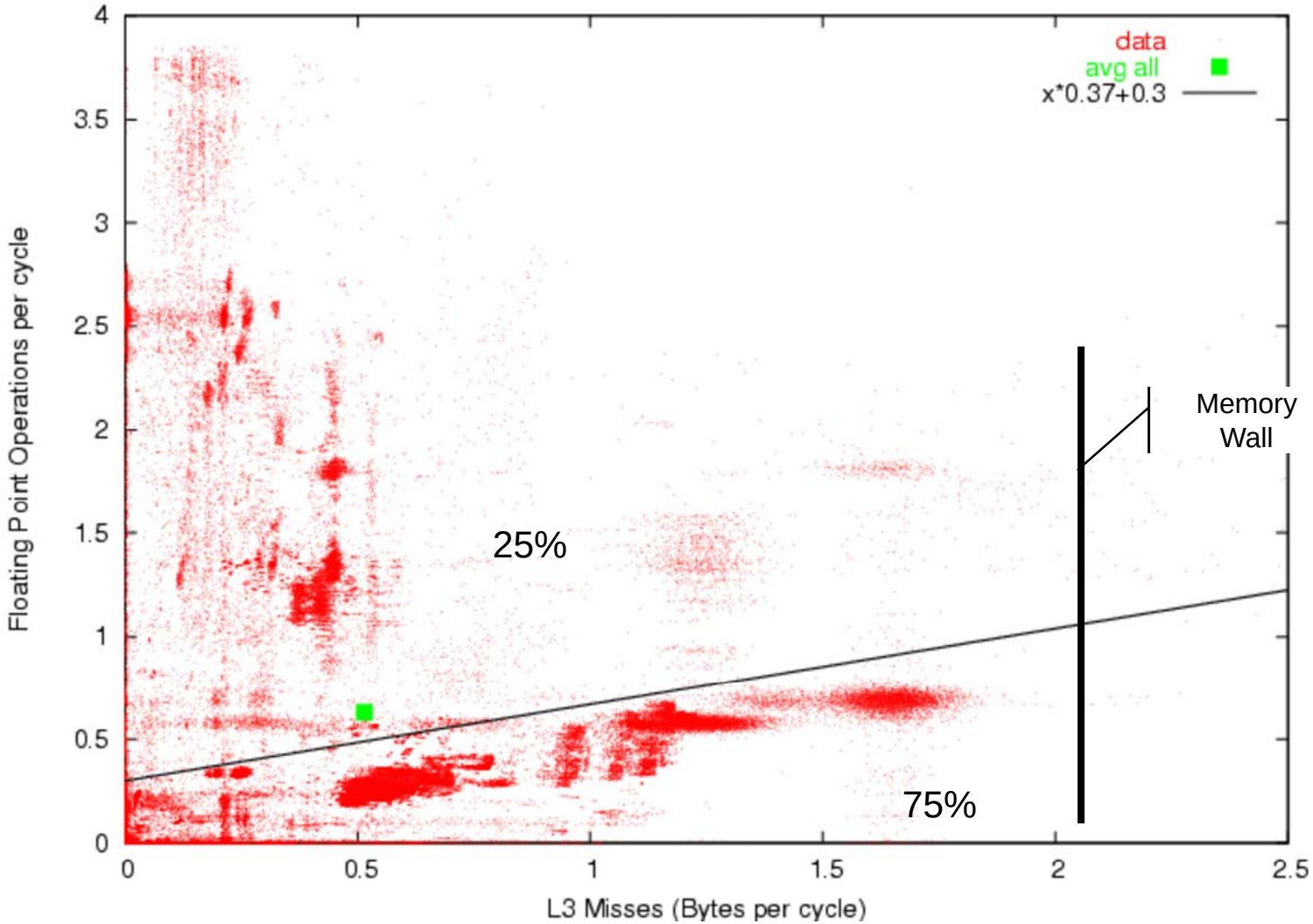
- **Severity:** High number of DCache misses per second
- **Criteria:** **>**2.00000E+05
- **Description:**
- This region shows a high number of cache misses per second and/ora high ratio of cache misses
- compared to the number of memory operations. This inhibits high performance in this region.
- **Advice:**
- Check the load/store behaviour. Try to pseudo-vectorize the code. Try to optimize for
- cache. Try to minimize the load (and store) instructions.
- If the code was already pseudo-vectorized, but a high number of cache misses remained,
- analyse the code for cache trashing. Array Padding or the use of *SOPTION COPYARG may be
- means to get rid of the thrashing.

- **Severity:** High number of DTBL misses per second
- **Criteria:** **>**3.00000E+02
- **Description:**
- This region indicates a large number of DTBL or ITBL misses. To reduce the number of DTLB or
- ITBL misses.
- **Advice:**
- Access data contiguously in memory. Use Large Pages via the compiler(/loader
- options +SBTLB and +BTLB.

Application Behaviour
Hitachi SR8000



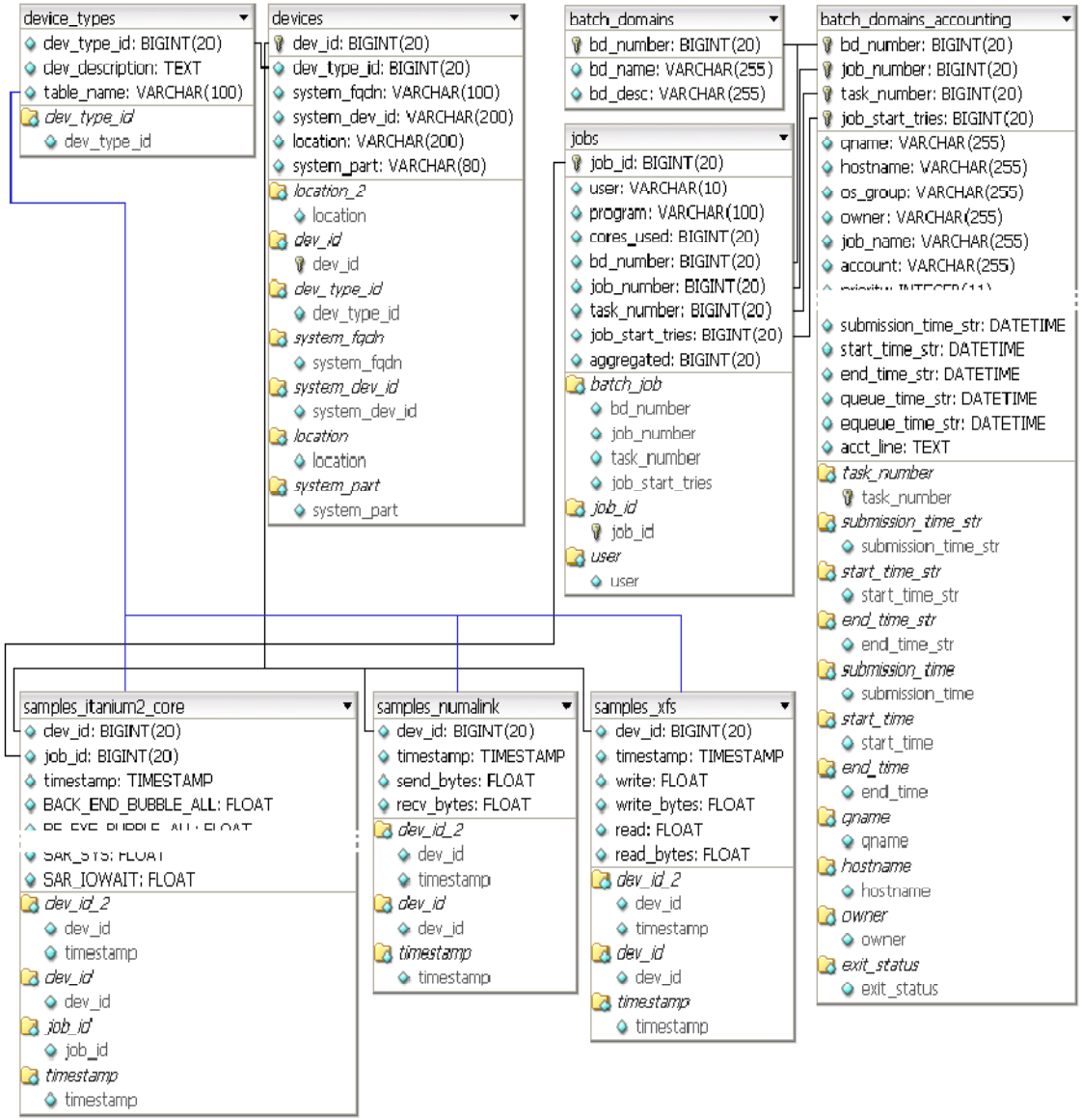
Application Behaviour
SGI Altix 3700



Current Approach:
Use of MySQL Database for HLRB-II (SGI ALTIX) and Linux-Cluster

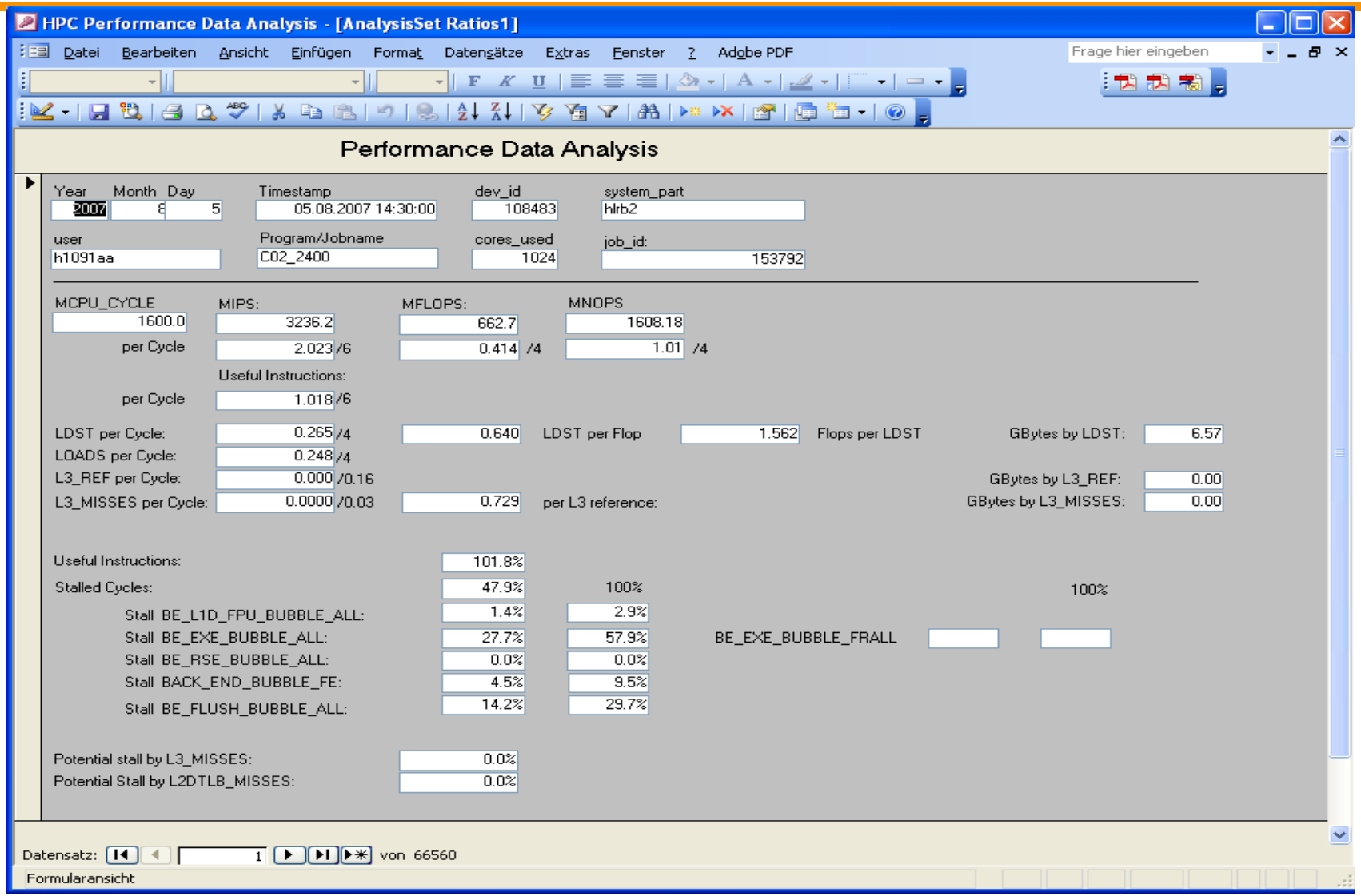
- **Why Database**
  - **Huge Amount of Data**
    - Indexing
  - **Transaction processing**
    - Data are consistent and inserted at the right place
  - **Joining with accouting / job information / user databases**
  - **Build-in Functions**
    - Grouping
    - Filtering
    - Sorting
  - **MS Access via ODBC**
    - Fast prototyping
    - *Ad hoc* complex questions
    - Graphics
    - Pivot Tables and Graphics

Database Approach for Linux-Cluster and 9728-Core Altix



- where the measurement was made, i.e. on which device (CPU, Network device, IO)
- Device description
  - GHz etc
- when it was made
- Job information
- Perf Counter Data
- Memory, sys, cpu

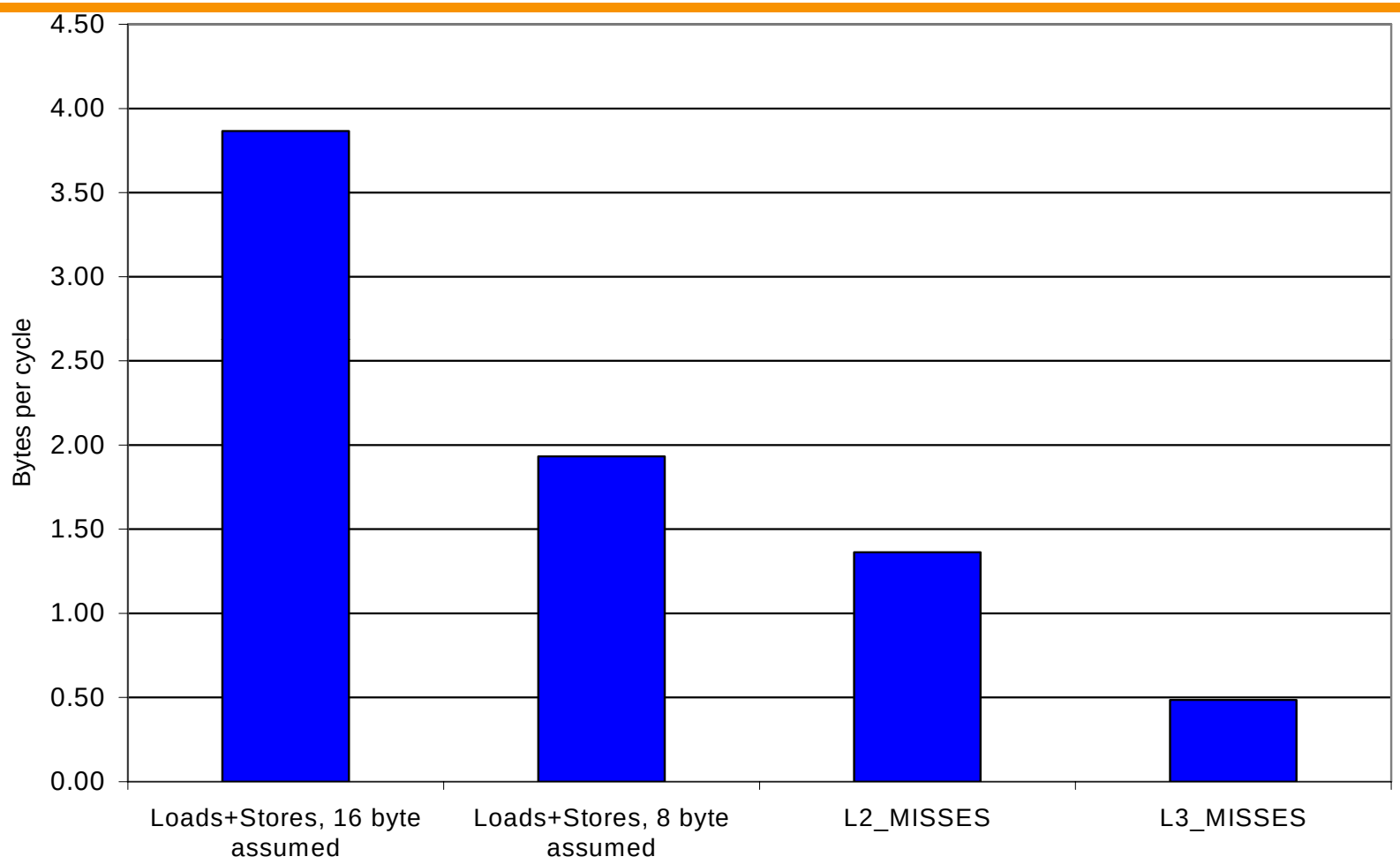
Interface for analysis



Enable user and support personnel to understand Memory Hierarchy

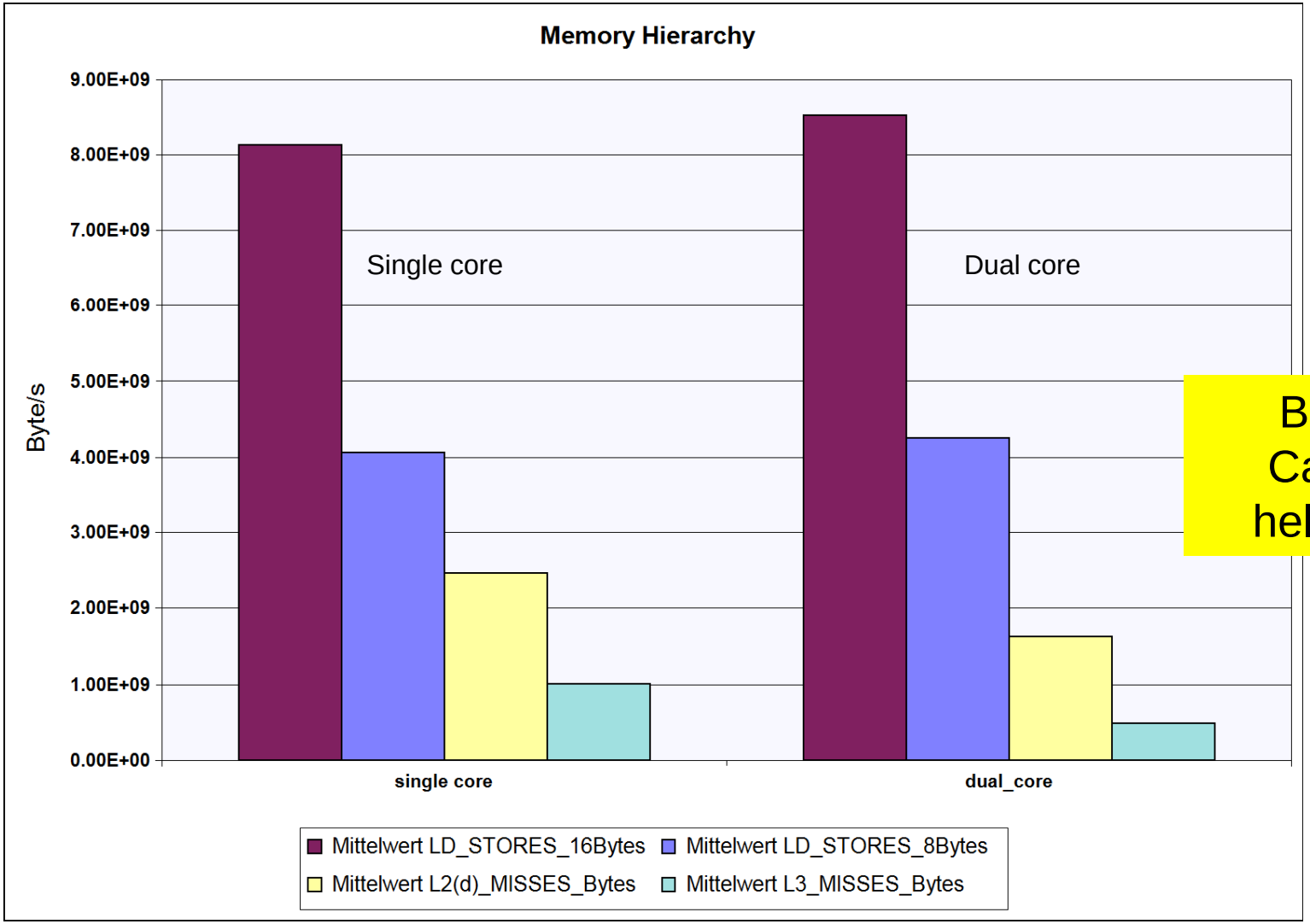
- Exploitation of the memory hierarchy of end-user codes gets more crucial with multicore
- Inclusion of communication within nodes and between nodes
  - Not generally provided as HW counters
- Messages Sizes – Bandwidth and Latency

Memory Hierarchy (SGI Altix 3700 Bx2) all applications (120000 samples)



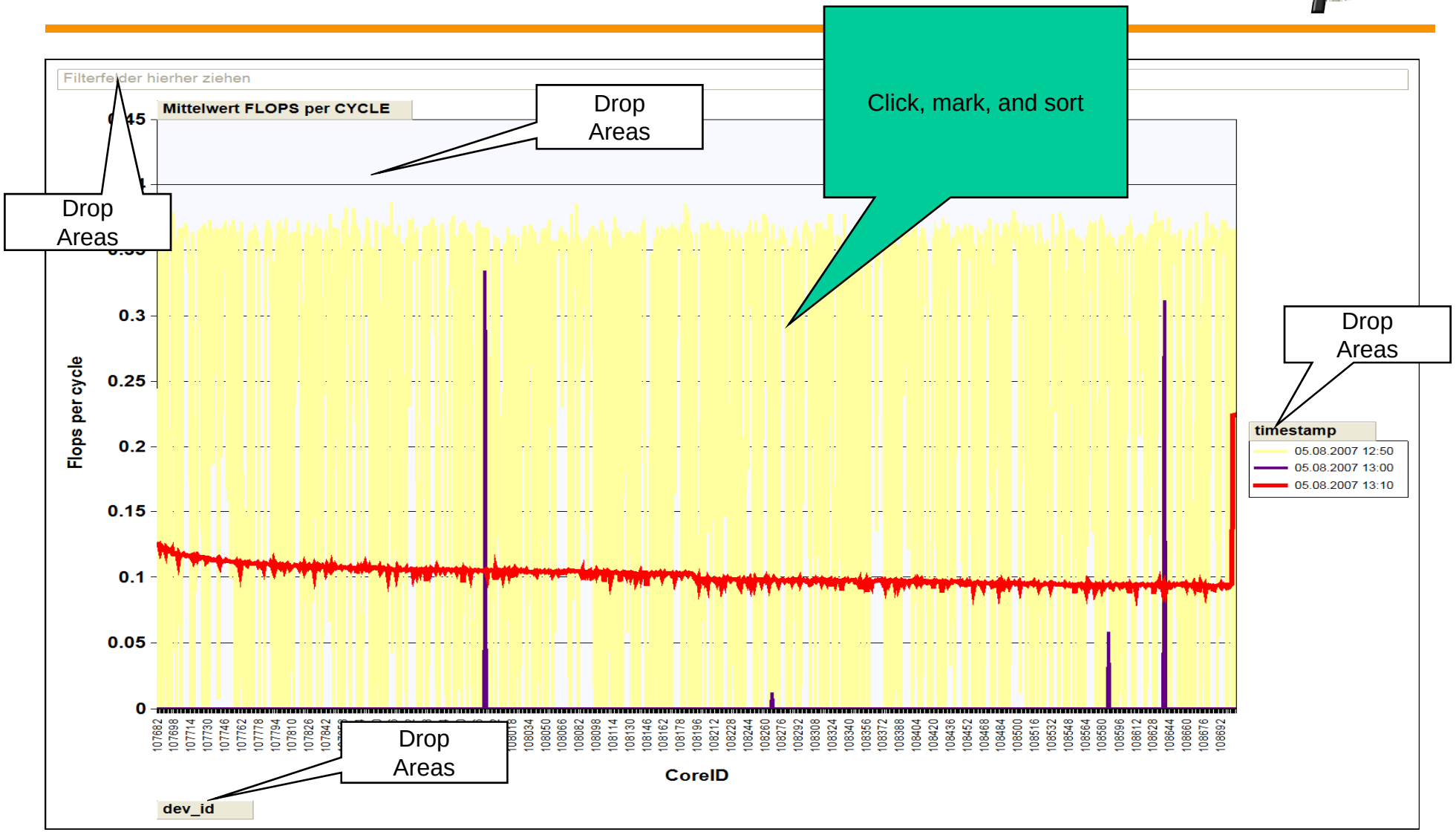
Register ⟷ L2 ⟷ L3 ⟷ Memory

Memory Hierarchy HLRB2 SGI Altix 4700
Single Core 6 MB L3 vs. Dual Core 9 MB L3

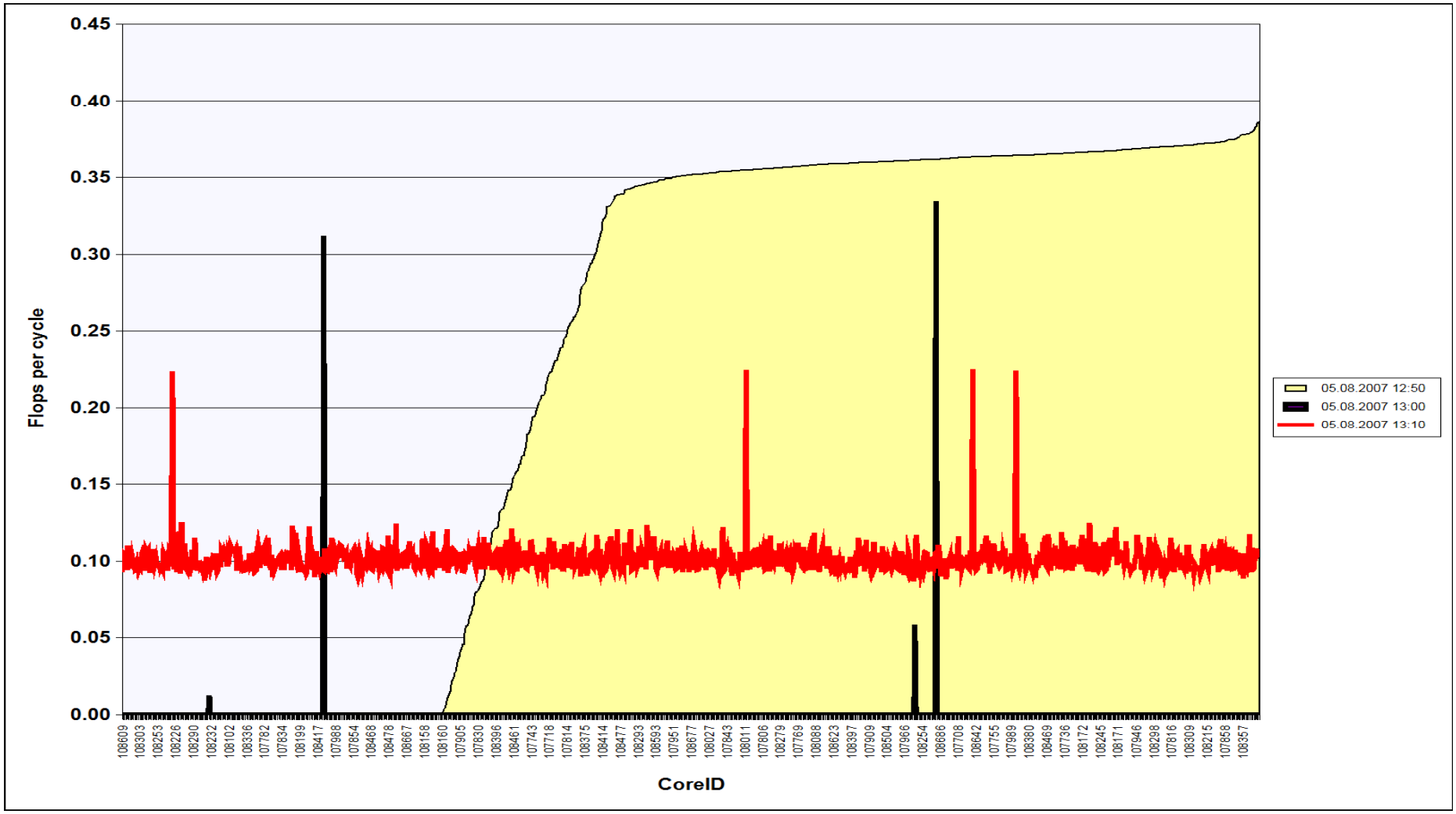


Bigger Caches help a lot

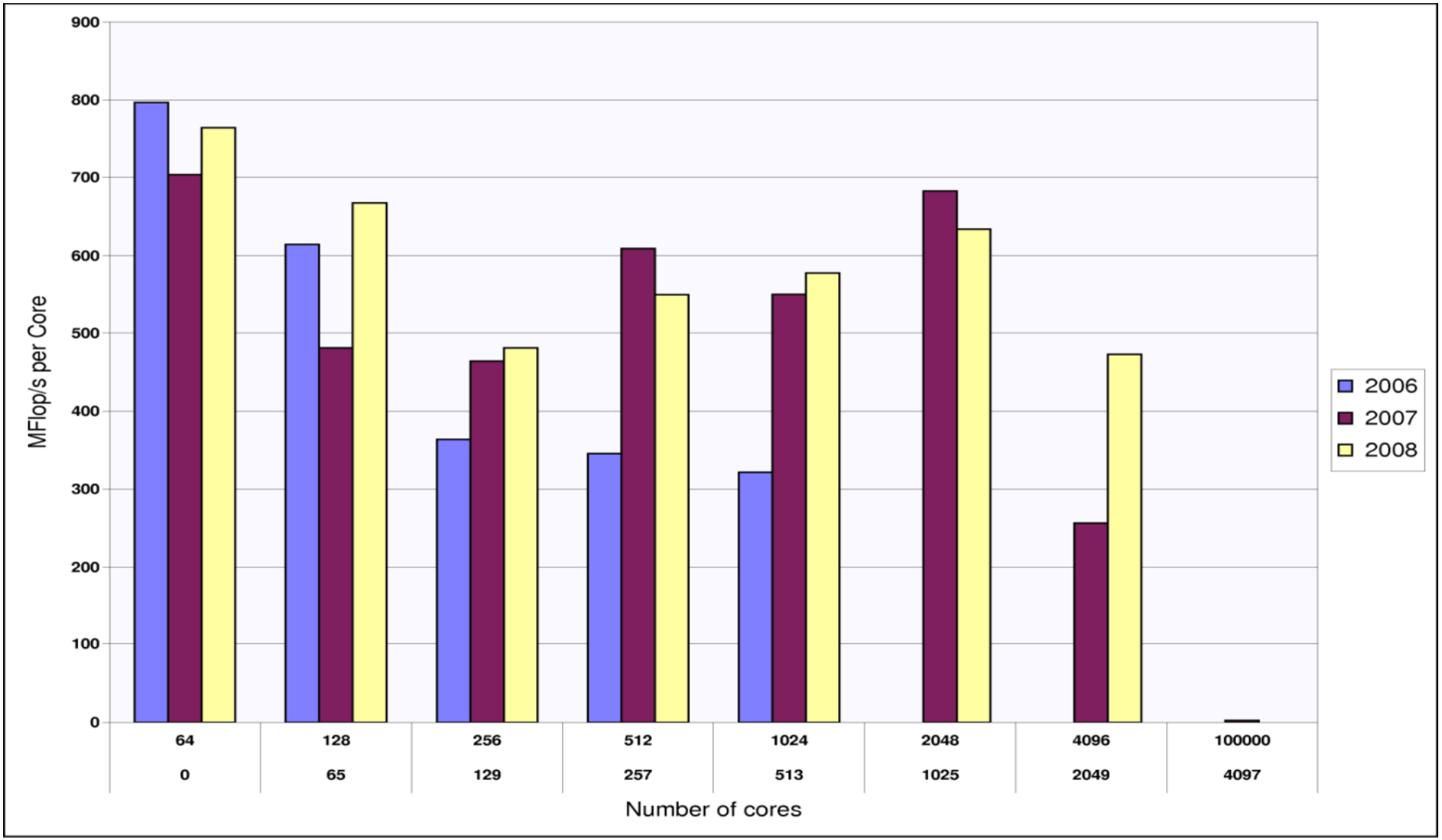
Needed: Ease Of Use to get insight
Example: Load Imbalance



Example: Load Imbalance



Distribution of Performance of all Jobs

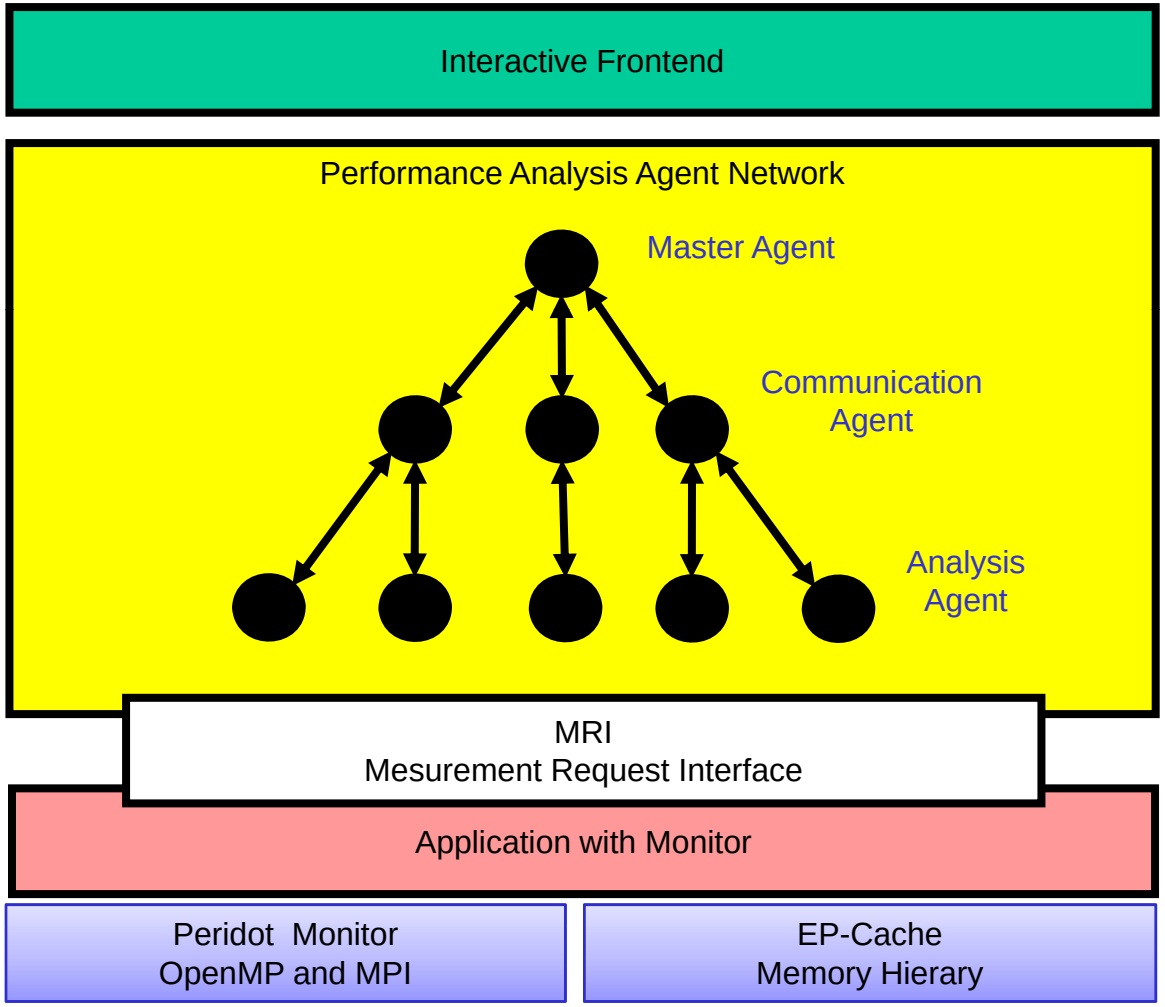


Monitoring
Problems known and seen

- **Storing and retrieving data for core count > 10000 is difficult**
  - volume of performance data
  - Analysis (on- and off-line) becomes a significant problem
- **Evaluation of alternate sampling techniques to reduce amount of data acquired and storage required,**
  - **e.g usage of statistical sampling instead of complete sampling on every resource**
- **Automatic filtering and aggregation of data**
- **Efficient methods for fast offload of monitored data to external devices (parallel file systems, SSD Devices) and asynchronous postprocessing of data (aggregation and long term storage) are needed**
  - **Archive only relevant information**
- **Monitoring techniques to offload monitoring task from system processors e.g., usage of service processors, hierarchy of agents**

- **Parallel Data Analysis/Postprocessing/Retrieval will be necessary**
  - **e.g. VAMPIR NG, SCALASCA**

Periscope Design
(Michael Gerndt et.al. TU Munich)



Agents perform a sequence of experiments for a program phase

Check potential performance problems based on the predefined ASL properties

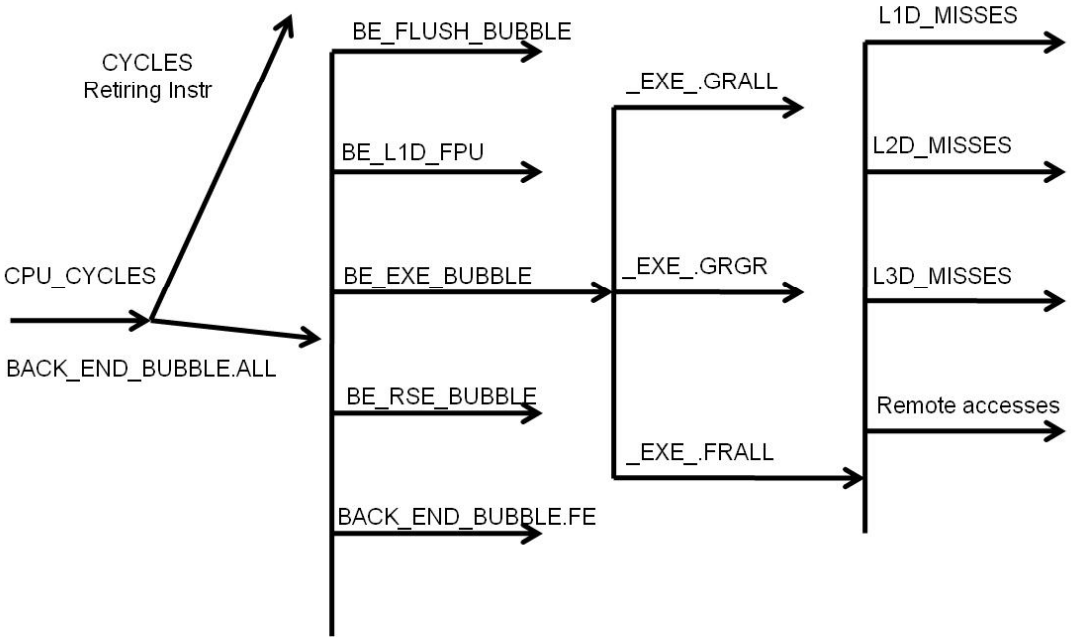
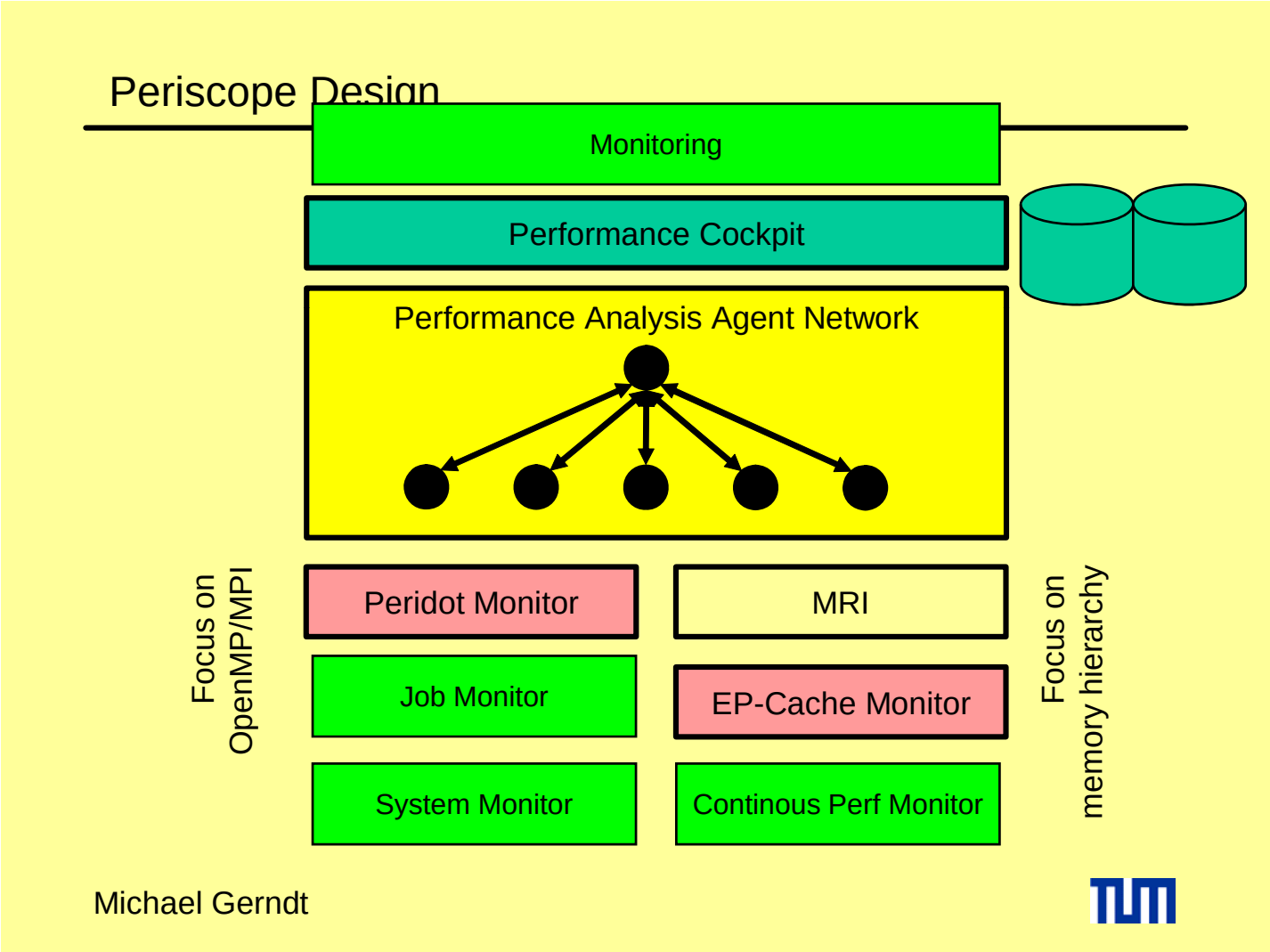
Searching for local and global performance properties
Multi-step analysis
Properties and data model based on ASL specification

Examples: Communication Costs

```
property communication_costs (
     Region r,
     Experiment e,
     Region rank_basis)
{
LET
  float cost = summary(r,e).CommTime;
IN
  CONDITION:   cost>0;
  CONFIDENCE:  1;
  SEVERITY:    cost/duration(rank_basis,e);
}
class RegionSummary {
   Region reg;
   float duration;
   float commTime;
   float ioTime;
   float syncTime;
   float idleTime;
   int nrExecutions;
   setof ProcSummary processSums;
}
```

Agent Search Strategies on IA64

- Predefined search strategies in repository
- Current phase-based search strategies
  - Region nesting-based refinement
  - Stall cycle analysis
- Refines in property hierarchy
  - LC2DMissRateInSomeThread -> UnbalLC2DMissRate
  - Cache miss ratio -> read/write miss ratio or local/remote references
- Each property defines a list of more precise properties.



Ideas for Future Monitoring System: Using The Periscope Frame Work