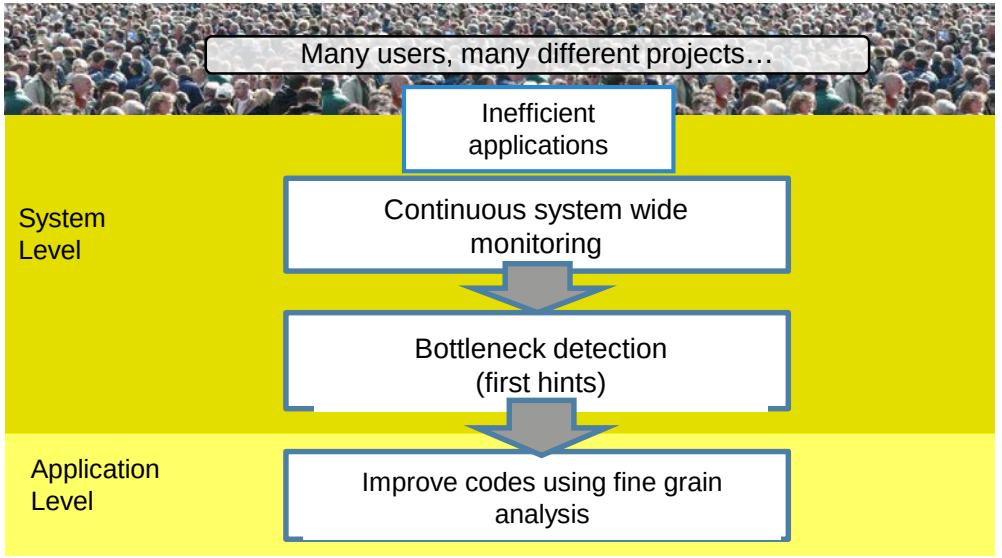
Korrelation von Daten, Methoden der Auswertung

W. Hesse, C. Guillen, C. Navarrete, M. Brehm

Motivation

Detektion ineffektiver und ineffizienter Applikationen auf System-Ebene

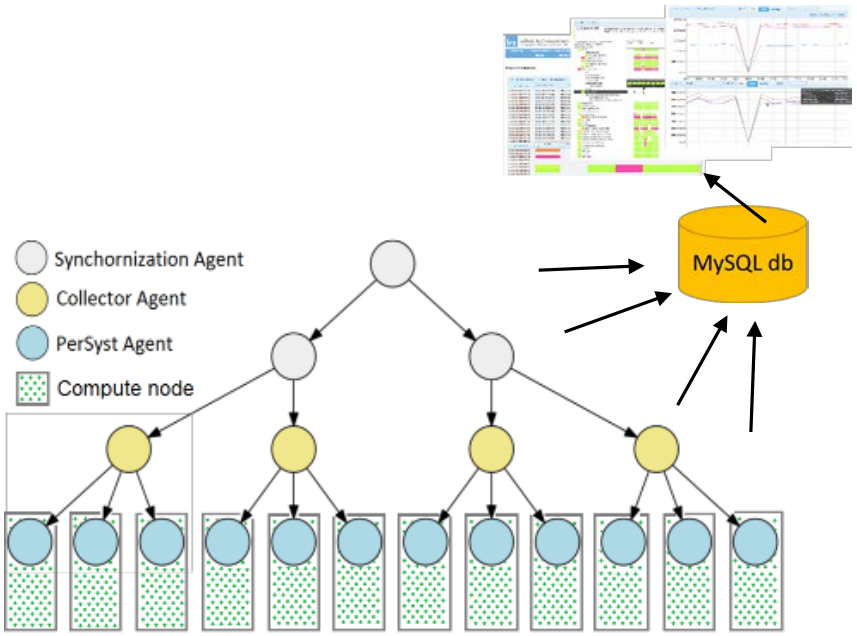


Problem: Performance Evaluation einer großen Anzahl von Anwendungen (Jobs) auf dem HPC-System

- Ziel des HPC-Rechenzentrum: Monitoring des gesamten Systems mit allen darauf laufenden Applikationen (Jobs)
  - Bsp. SuperMUC: > 20,000 Jobs pro Monat mit 16, …,148576 Cores (Ø 300)
  - Pro Core und Messung mehrere Events (z.B. Perf. Counter)
  - Mehrere Messungen (abhängig von Job-Länge + Messabstand)
- Hohe Anforderungen an Datenspeicherung sowie -Auswertung

LRZ: PerSyst-Monitoring

- Verteilte/hierarchische Architektur:
  - PerSyst-Agent (LIKWID-Messung)
  - Collector-Agent (Aggregation)
  - Synchronisations-Agent
- Keine Instrumentierung des Anwender-Codes („läuft nebenbei“)
- Systemweites Monitoring + Analyse
- **Selektives Monitoring** mittels Strategie (Maps)
  - Datenreduktion & Detektion von Bottlenecks (Teil 1)

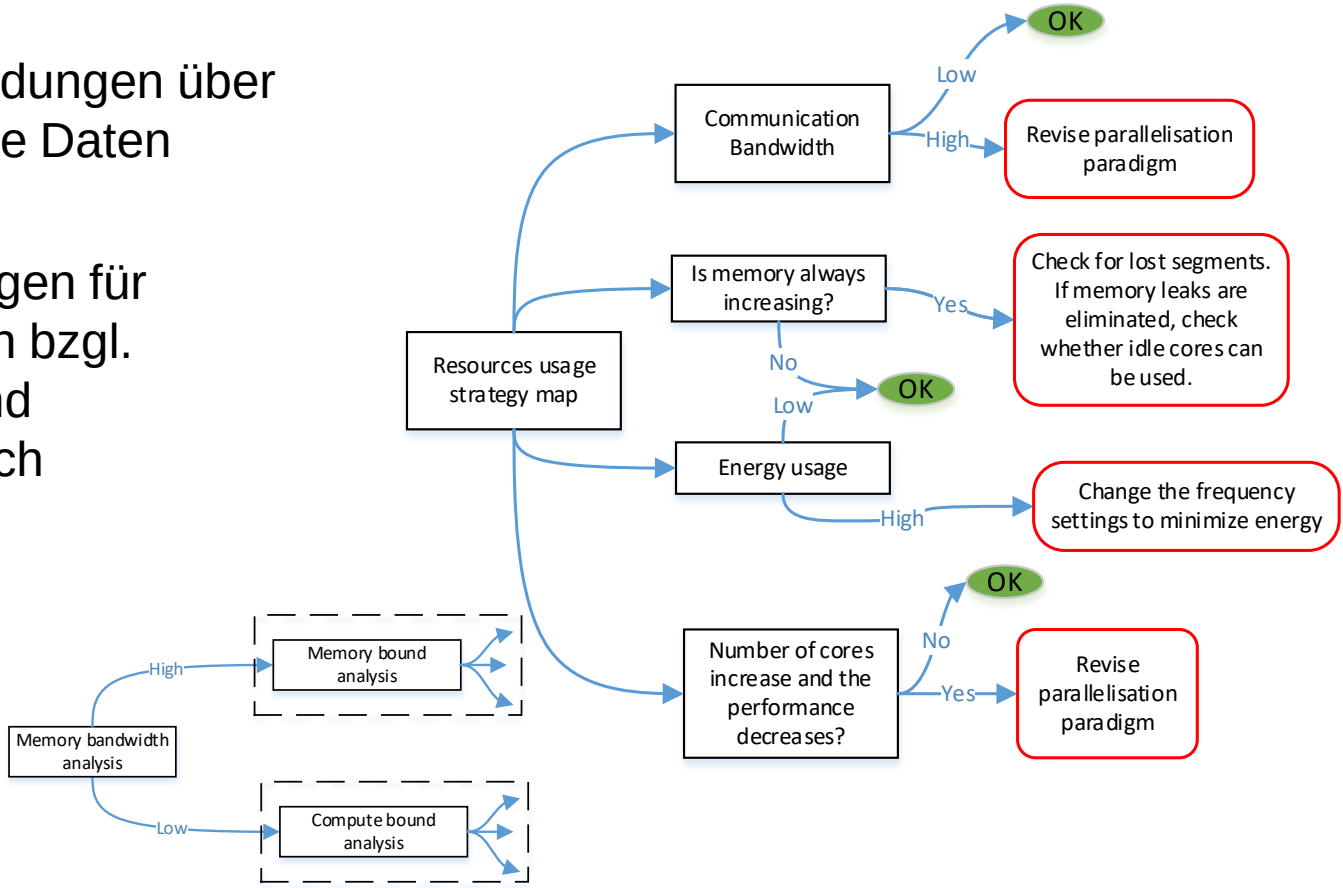


- **Statt Roh-Daten** werden **Verteilungen** der Performance-Muster pro Job **gespeichert (Perzentile, Average, Max, Min)**
  - Korrelation der system-weit gewonnenen Performance/Energie Daten mit den Informationen des Batch Scheduler
  - Datenreduktion & Detektion von Bottlenecks (Teil 2)
- MySQL-DB als Schnittstelle zur Visualisierung

PerSyst: Exkurs Selektives Monitoring

Selektives Monitoring durch Verwendung von Strategien

- Online Entscheidungen über einzusammelnde Daten
- Gibt Empfehlungen für Verbesserungen bzgl. Performance und Energieverbrauch



PerSyst: Exkurs Selektives Monitoring

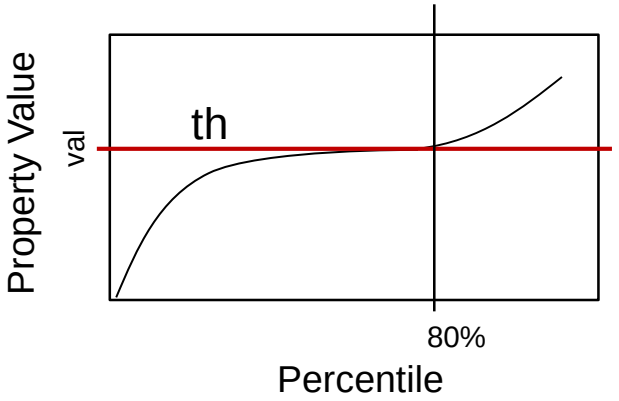
Selektives Monitoring durch Verwendung von Strategien

- *Property*-Werte (Funktioneller Zusammenhang zw. gemessenen Performance-Metriken)

  Bsp.: Floating Point Precision usage =

  Single Precision flops / Double Precision flops

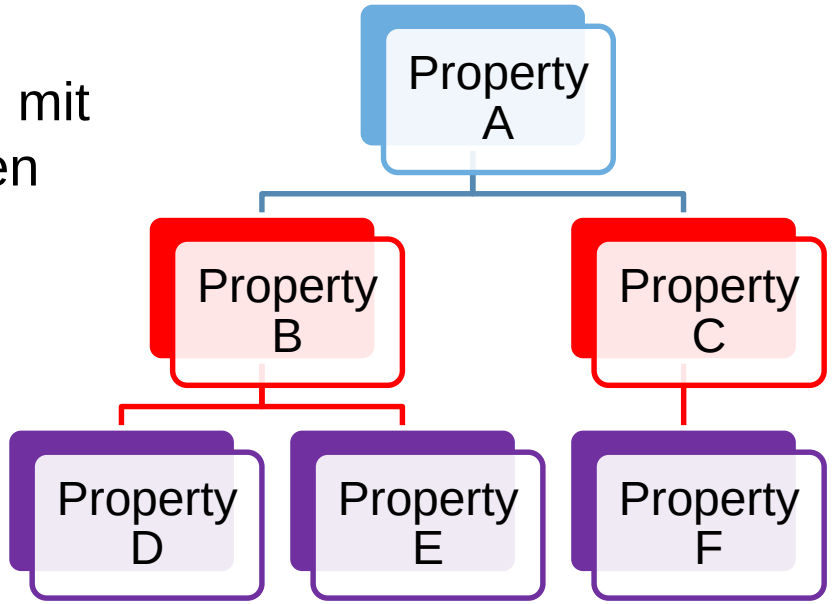
- *Schwellen aus*:
  - Hardwarecharakteristik
  - Benchmark-Untersuchungen
  - Statistischer Betrachtungen alle auf dem HPC-System laufenden Applikationen
    -> ständige Aktualisierung
- Berechnung der Schwere einer Property-Werte-Überschreitung (*Severity* (**0<s<1**))



PerSyst Datenselektion/-Reduktion

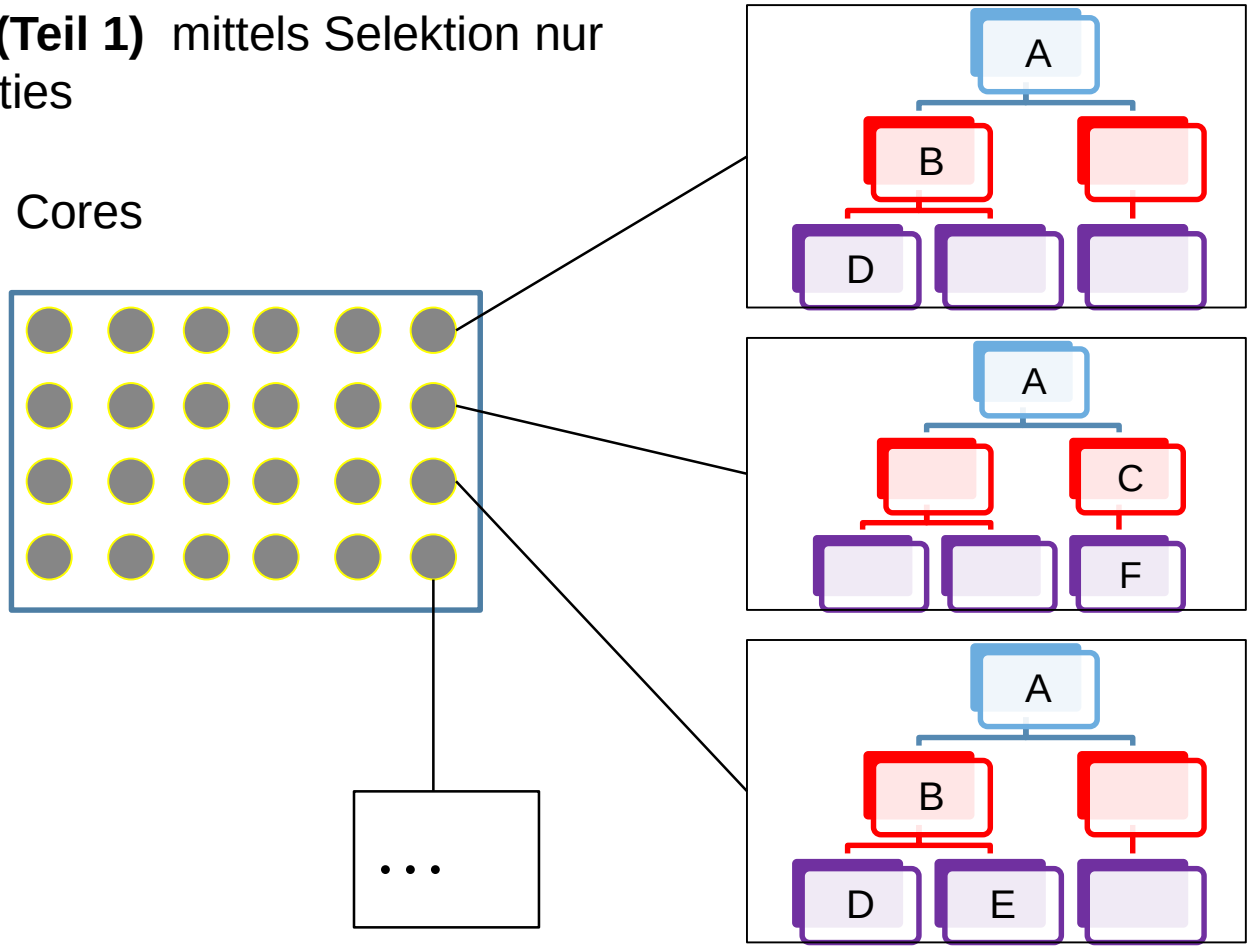
Datenreduktion & Detektion von Bottlenecks **(Teil 1)**

- Strategie „=“ Property-Baum
- Traversierung bei Properties mit Severity **s>0**
- „Top down“ Verfeinerung mit spezielleren Informationen



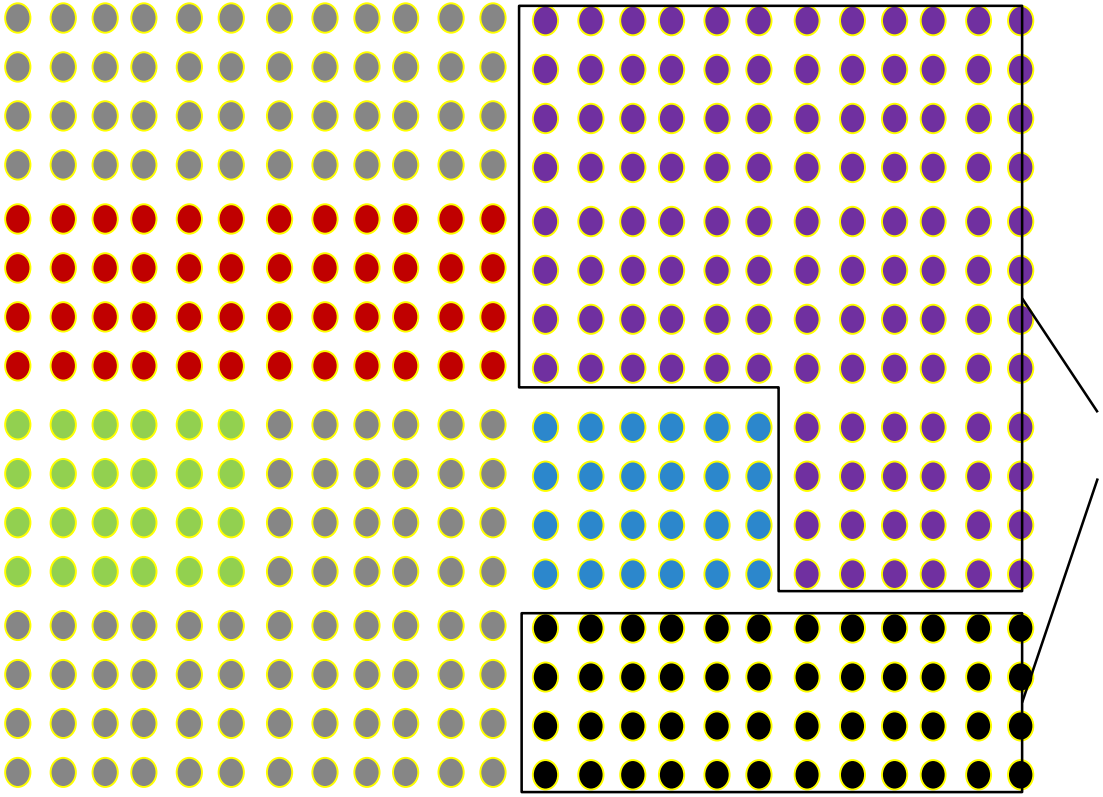
PerSyst Datenselektion/-Reduktion

Datenreduktion (Teil 1) mittels Selektion nur relevanter Properties



PerSyst Datenselektion/-Reduktion

Datenreduktion (Teil 2) mittels Abspeicherung der statistischen Verteilung



der Properties pro Job (unabhängig der Job-Größe), repräsentiert durch 12 feste statistischen Parameter:

- Mittelwert
- 11 Perzentile (inkl. „Minimum“ und Maximum)

sowie der Core-Anzahl.

-> Zurückgewinnung der Roh-Werte im statistischen Sinne (Schätzung der Roh-Werte)
-> Datenreduktion um ca. 90%

PerSyst Exkurs Perzentile

Daten:

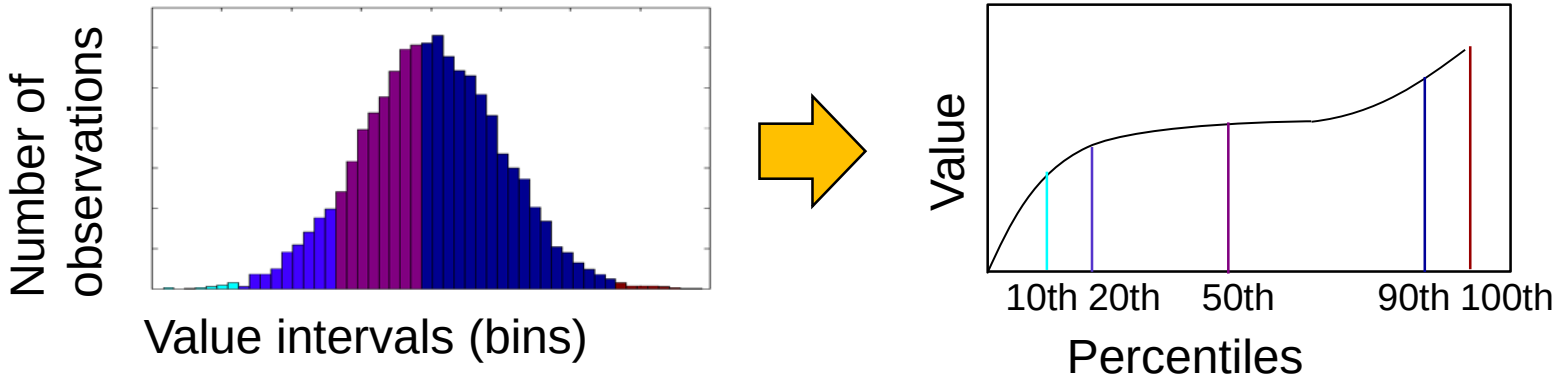
7	8	10	7	6	3	6	7	12	8	8	9	4	8	9	8	5	11	10	7

Sortierung:

3	4	5	6	6	7	7	7	7	8	8	8	8	8	9	9	10	10	11	12

Ermittlung der Perzentile:

0th	10th		20th		30th		40th		50th		60th		70th		80th		90th		100th
3	4	5	6	6	7	7	7	7	8	8	8	8	8	9	9	10	10	11	12



Weitere Bemerkungen Diskussionspunkte:

1. Datenbanksystem (verteilte NoSQL vs. SQL/Relationale)
   - Datenmenge
   - Performance bzgl. Schreiben/Lesen
   - Netz-Traffic
   - Anwenderfreundlichkeit, Kosten, etc.
2. Anwender/Nutzer (System-Admins, (HPC) Benutzer, PI)
3. Roh-Daten vs. aggregierte/profile Daten vs. statistische Daten
4. Messintervall (Sampling-Rate)
   - Aller 0.1,1,2,5,10, … Minuten oder viel öfters?
   - Aussagekraft, Interpretation, Fehler(Wahrscheinlichkeit)
   - Interpolierte Zeitreihen
5. Visualisierung:
   - Siehe Messintervall
   - Hinweise vs. „signifikante“ Aussagen
6. ….