DKRZ

DEUTSCHES
KLIMARECHENZENTRUM

Ansätze zum Monitoring hierarchischer E/A-Systeme

Julian Kunkel

ZKI-Arbeitskreis Supercomputing

Agenda

- Motivation
- Job-Statistiken
  - Verfügbare Metriken
  - Technische Umsetzung
- HPSS-Monitoring

Ziele

- Transparenz!

➢ Unter minimalen Leistungseinbußen erfassen:

- Job-Statistiken:
  - Prozessor
  - E/A
  - Netzwerk
  - (Batch-System), (Betriebssystem)
- HSM-Nutzung
  - Anwender- & projektbezogen

➢ Lebenszyklus der Daten

Motivation

- Analyse des Job-Mixes & Datenlebenszyklus erlaubt:
  - Aufdecken von Ineffizienzen
    - RZ: Benutzerrückmeldung, Unterstützung
    - Wenig Leistung => Analyse mittels Spurbasierte Werkzeuge
    - Versionsübergreifende Leistungsanalyse von Programmen
  - Hardware-Probleme detektieren
  - Erstellung von Reports
    - Planung der nächsten Rechnergeneration
  - Abrechnung

Verfügbare Metriken

DKRZ
DEUTSCHES KLIMARECHENZENTRUM

Prozessor

- IBM POWER6
  - 6 Performance Monitor Counters, 553 Events, 202 Groups
  - 2 fest: PM_INST_CMPL, PM_RUN_CYC
- Relevant:
  - FLOPS
  - Prozessorauslastung
  - Instructions per Cycle
  - Cache-Nutzung
  - …

Netzwerk

- Infiniband Host-Channel-Adapter
  - PortCounters speichern:
  - Anzahl Pakete
  - Anzahl Bytes (Senden, Empfangen)
  - Anzahl Fehler

I/O-Subsystem

- Eingabe/Ausgabe in GPFS
  - Zähler für Statistiken auf Knoten
    - Für jedes Dateisystem und Knoten aggregiert
    - Anzahl open, close, reads, writes, readdir, inode update
    - Datenmenge Lesen/Schreiben in Bytes
    - Werden immer geführt
    - Histogramme optional
      - Gruppiert Bereiche nach Größe und Latenz

GPFS-Histogramme

```
size range                0 to          512 count          539
   latency range          0.0 to          1.0 count          494
   latency range          1.1 to         10.0 count           29
   …
   latency range        100.1 to        200.0 count            2

size range              513 to         1024 count           85
   latency range          0.0 to          1.0 count           81
   …
   latency range        100.1 to        200.0 count            1
…
```

Technische Umsetzung

Umsetzung

- Alternativen:
  - LoadLeveler Prolog- und Epilog-Skripte anpassen
    - Existierende Zähler abfragen und Werte speichern
  - Mpiexec (poe) wrappen
    - Vorteil: Information für jedes parallele Program
- Automatische Instrumentierung vs. Spurwerkzeuge
  - Werkzeuge beeinflussen sich gegenseitig
    - ➢ LoadLeveler Queue ohne Monitoring zur Leistungsanalyse

Datenquellen

- Prozessor
  - Hardware Performance Monitor (HPM) Toolkit
    - hpmcount
      - Multiplexing von Ereignissgruppen und Normalisierung
      - Ausgabe der Resultate in Datei(en)
- Netzwerk
  - Auf jedem Knoten IB-HCA PortCounters auslesen
- E/A-Subsystem
  - Mit mmpmon Zähler auslesen

Datenerfassung

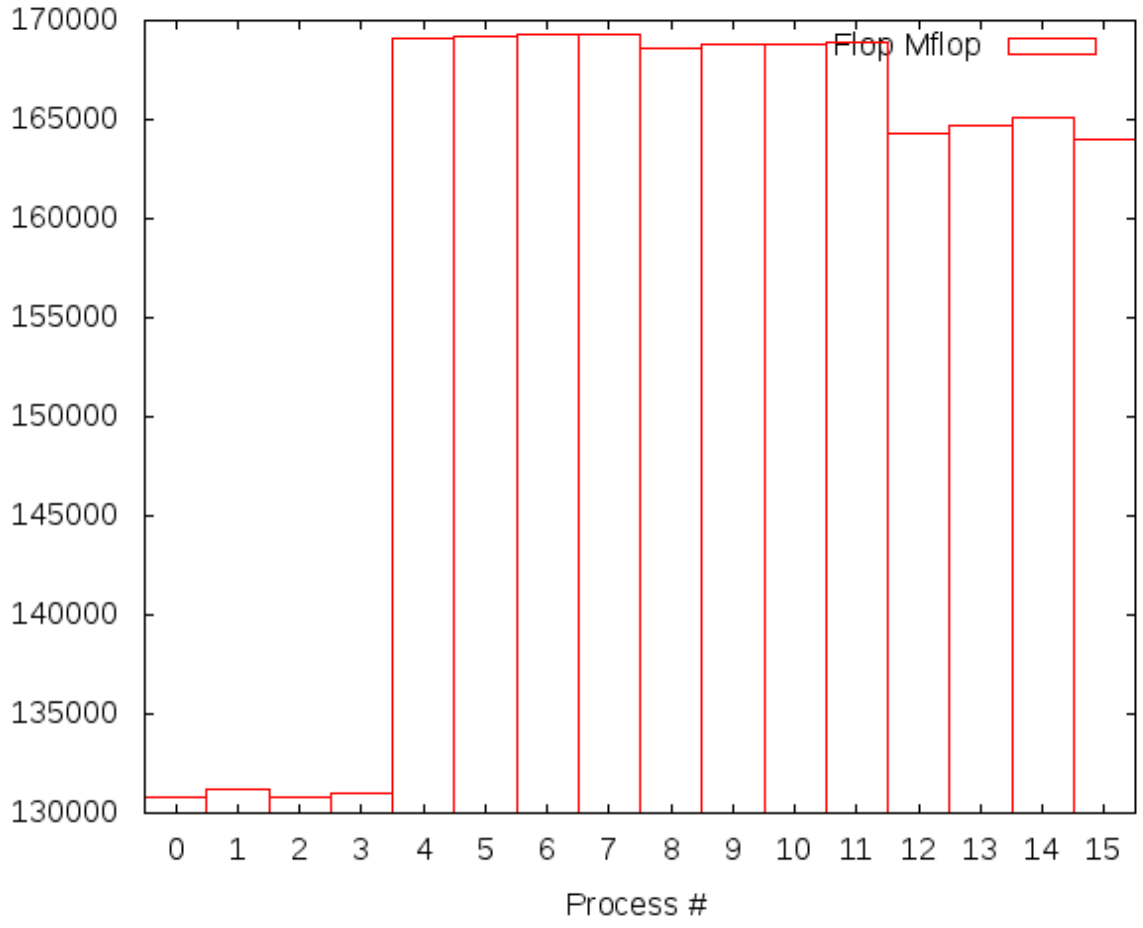
- Wrapper-Skript
  - Starten von parallelen Jobs durch Wrapper Skript

    <counterAbfragen>

    poe hmpcount *<options> <program> <parameter>*

    <counterAbfragen>
  - Parser für die Ausgaben in LoadLeveler Epilog-Skript
  - Reduktion der gesammelten Daten auf Kennzahlen
    - Ausgabe für den Benutzer und für Administrator
    - Ergebnisse nachträglich im Detail analysieren (pro Prozess…)

Hardware Zähler pro Prozess



Vorgehen

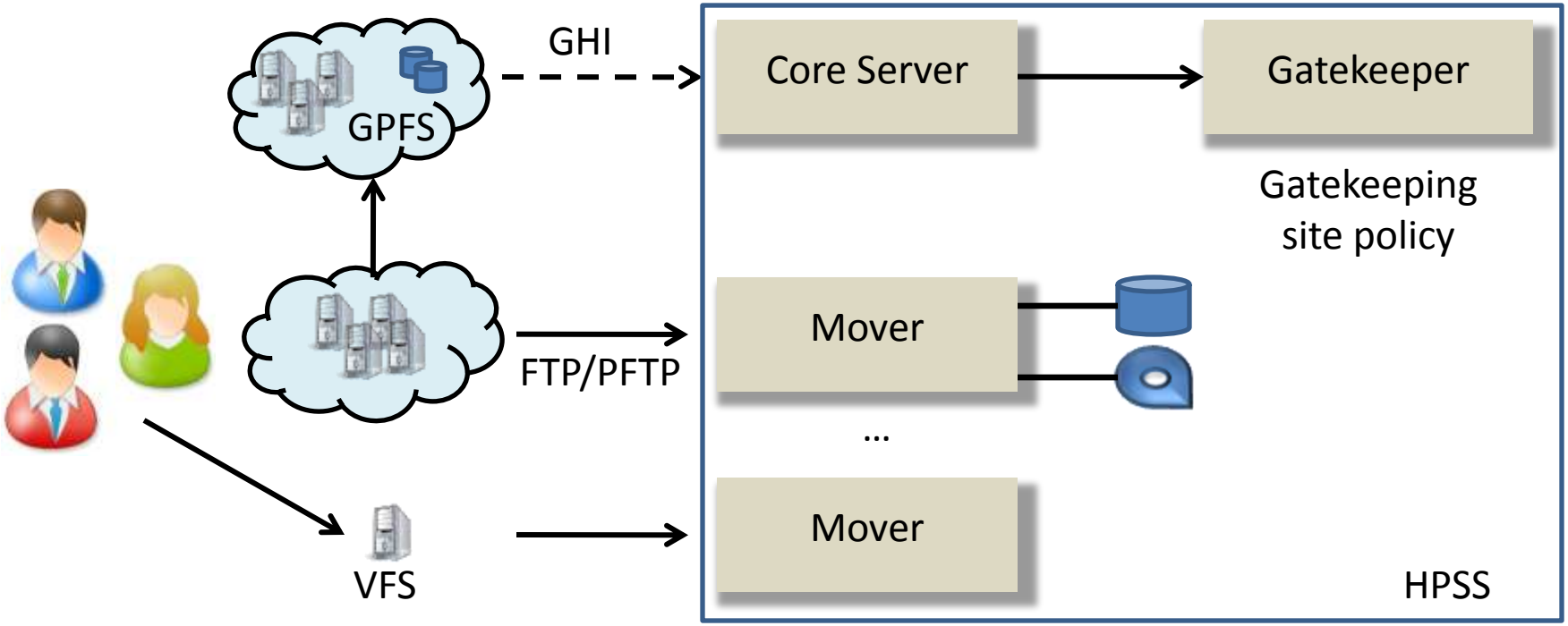
- Phase 1: Infrastruktur für existierende Werkzeuge
  - Erfassen von CPU, Netzwerk, E/A-Statistiken in Dateien
  - Zusatzlast durch Werkzeuge überprüfen
- Phase 2: Robustheit / Leistungsfähigkeit
  - Datenbank-Design, zu erfassende Daten anpassen
- Phase 3: Aufzeichnen der GPFS-Server-Information
  - Zusätzlich zur Client-Information speichern

HPSS

Abbrechnung Ziele

- Migration von Dateien:
  - Kosten je angefangenes GByte berechnen
- Löschen von Dateien
  - Anreiz für Löschen schaffen
  - Aber weiterhin Platzbedarf auf Band
    - Bis Band umkopiert wird
  - ➢ ½ Speicherplatz für Projekt wieder freigeben
- RZ: Nutzerunterstützung - redundante Daten entdecken

HPSS Nutzungsschema



Aspekte

- Benötigte Informationen:
  - Veränderungen der Metadaten
  - Migration/Staging/Purge, lateral Move
  - Objekt (Info verfügbar vs. ID -> DB Abfrage)
- Verfügbarkeit: zentral vs. dezentral
- Umfang der Code-Modifikation
  - Verlust des Supports von IBM
  - 1.117.128 LOC (*.c Dateien), Core: 250.000 LOC

Monitoring Alternativen

- Serverseitige Protokolle(FTP-Daemon, Mover, …)
  - Metadaten Veränderungen werden nicht aufgezeichnet
    - Löschen von Dateien?
- Server-Instrumentieren
  - Supportverlust bei Quellcode-Modifikation
- Gatekeeper Site-Policy nutzen
  - Erfasst nur: Create, Open, Close, Stage
- Clients instrumentieren

Serverseitige Protokolle

- Abbrechnung der Nutzer mit Protokollen möglich

```
# Realmid AcctId  COS       0    #Accesses          #Files          Length  (COS)
# Realmid AcctId  COS  SClass    #Accesses     Transferred (SClass)
       0     178    1       0            0                 0               0
       0     178  322       0         3926              2703      5174031270483
       0     178  322      33         3926      12122156232666
```

  - Aber nicht für Projekte

- Zugriffsprotokolle enthalten:

  Client, op. Dauer (Open), Dateigröße & Name, Benutzer, Transfer-Typ, Class-of-Service

Existierende Analysen

- Log-Information wurde bereits genutzt:
  - In Datenbank integrieren (von IBM am DKRZ)
    - Pro Nutzer ausgewertet:
      - Datengrößen, # Dateien …
  - In USA:
    - Workload-Charakterisierung
    - Zeit bis Dateien gelesen wurden (Staging, Cache)
  - Aber keine Metadaten, Lebenszyklus der Daten

Gatekeeper

- Erweiterung mittels *shared library* vorgesehen
- Wird vom Core Server aufgerufen
- Erlaubt Abweisen von Anfragen und Monitoring
- API für open, close, create, staging
- Protokolle an einer Stelle

Client-API

- Wohldefiniertes Interface (viele Fkt)
- Viele Clients => Datenquellen
- Wrapper für die Funktionen schreiben
  - Umbenennen der normalen Funktionen
  - Hinzufügen von Wrapper Bibliothek (vgl. PMPI)
  - ➢ Überschaubare Änderungen
  - Alle Clients neu kompilieren
- Dynamisch instrumentieren z.B. mit DPCL

Serverseitige Instrumentierung

- Alle Anfragen gehen durch Core Server
- Objekte werden per IDs referenziert
- Lebenszyklus & MD-Operationen verfolgbar
- Für jede Anfrage instrumentieren
  - Zeit/Parameter/Rückgabe in Text aufzeichnen

Datenlebenszyklus

Lebenszyklus von Dateien aus Protokollen rekonstruieren

- Mustererkennung und Gruppenbildung