Existierende Backend-Komponenten

Jörg Benke, Hinnerk Stüben

Universität Hamburg, Regionales Rechenzentrum

FEPA Workshop *Job-spezifisches Performance-Monitoring*
Regionales Rechenzentrum Erlangen (RRZE)

20. Juli 2017

Gefördert durch die Deutsche Forschungsgemeinschaft (DFG)
KO 3394/14-1, OL 241/3-1, RE 1389/9-1, VO 1262/1-1, YA 191/10-1

Inhalt

Inhalt

Überblick

- Diskussion von *Deliverable 2.1* aus dem ProfiT-HPC Projekt
  - *Concise overview of performance metrics and tools*

`https://profit-hpc.de/wp-content/uploads/2017/04/profithpc_deliverable_2-1_final.pdf`

Ziele von ProfiT-HPC

- Automatisierung
  - der Erzeugung von Laufzeitprofilen (idealerweise)
  - der Erstellung von *verständlichen* Performance-Analysen

Methodisches

- Monitoring
  - Linux-Ebene
  - Ziel: vernachlässigbarer Overhead
- Profiling
  - Anwenderprogrammebene
  - Overhead kann variieren (Sampling vs. Tracing/Instrumentierung)
- Auflösung der ermittelten Performance-Daten
  - zeitlich aufgelöst vs. Summen
  - räumlich aufgelöst (Knoten, cores / Prozesse, Threads) vs. Mittel- und Extremwerte
  - Software-Granularität: Job-Skript (einzelne Kommandos) und Binärprogramm (Unterprogramme)

Inhalt

Diversität

- Software-Bereitstellung
- Übersetzungsumgebungen
- Batch-Systeme und Job-Typen
- I/O und Dateisysteme

Software-Bereitstellung

- durch den Betreiber
  - Übersetzung von Paketen
  - Entpacken von Binärdistributionen
- durch den Nutzer
  - selbst geschriebene Programme
  - Übersetzung von Paketen

Übersetzungsumgebungen

- Compiler (GNU, Intel, PGI, . . .)
- MPI-Implementierungen (Intel, Open-MPI, MPICH, . . .)
- CUDA

Batch-Systeme und Job-Typen

- Batch-Systeme
  - Slurm, Torque, LSF, ...
- Parallelität
  - seriell
  - parallel: OpenMP, MPI, hybride
- Knotennutzung
  - dediziert vs. gemeinsam
  - ein vs. mehrere Knoten

I/O und Dateisysteme

- lokale Festplatten
- globale Dateisysteme
  - NFS
  - parallele Dateisysteme (BeeGFS, GPFS, Lustre, . . .)

Inhalt

Ergebnisse aus D2.1

- Eigenschaften der untersuchten Tools
  - Art der Metriken
    - CPU, Hardware-Counters, Memory, MPI, I/O
  - Messmethoden
    - Sampling, Tracing
  - operative Aspekte
    - Einfachheit der Bedienung, Automatisierbarkeit, Overhead, Verfügbarkeit
- Beschränkung auf Open-Source-Tools

Ergebnisse aus D2.1

- Bewertung im Hinblick auf die Verwendbarkeit in ProfiT-HPC:
  - – The tool in this respect does not support or is not relevant to our use case.
  - ⋆ The usage of the tool regarding this criteria is possible but not recommended.
  - ⋆⋆ Regarding this measure, using this tool is practical.
  - ⋆⋆⋆ This tool is an excellent candidate to use with respect to our use cases.
- Keine Unterscheidung zwischen *Monitoring* und *Profiling*.

Untersuchte GNU/Linux Tools

- time
- GNU time
- strace
- ltrace
- ftrace
- perf
- gprof
- procfs
- sysstat

Untersuchte GNU/Linux Tools

	General	HW Counters	Memory	File System	MPI	Sampling	Tracing	Ease of Use	Automatability	Overhead	Availabi
time	⋆	–	–	–	–	–	–	⋆⋆⋆	⋆⋆⋆	⋆⋆⋆	⋆⋆⋆
GNU time	⋆⋆	–	⋆	⋆	–	–	–	⋆⋆⋆	⋆⋆⋆	⋆⋆⋆	⋆⋆⋆
strace	⋆	–	⋆	⋆	–	–	⋆⋆	⋆⋆	⋆⋆	⋆	⋆⋆⋆
ltrace	⋆⋆	–	⋆	⋆	⋆	–	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆⋆
ftrace	⋆⋆	–	⋆	⋆	–	–	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆⋆
perf	⋆⋆⋆	⋆⋆	⋆	⋆	⋆	⋆⋆	–	⋆⋆	⋆⋆	⋆	⋆⋆⋆
gprof	⋆⋆	–	–	–	⋆	⋆⋆	–	⋆	⋆⋆	⋆	⋆⋆⋆
procfs	⋆⋆	–	⋆⋆	⋆⋆	–	⋆	–	⋆⋆⋆	⋆⋆⋆	⋆⋆	⋆⋆⋆
sysstat	⋆⋆	–	⋆	⋆⋆	–	–	–	⋆⋆	⋆⋆	⋆⋆	⋆⋆⋆

Untersuchte allgemeine Tools

- Darshan
- gperftools
- HPCToolkit
- IPM
- LIKWID
- MAQAO
- memP
- MPE
- mpiP
- MUST
- ompP
- OProfile
- Open|SpeedShop
- PAPI
- Paradyn
- PMPI
- Score-P
- Slurm
- TAU
- Valgrind

Untersuchte allgemeine Tools

	General	HW Counters	Memory	File System	MPI	Sampling	Tracing	Ease of Use	Automatability	Overhead	Avail-
Darshan	★★	–	–	★★★	★★	–	★★	★★	★★	★★★	★
gperftools	★★	–	★★★	–	–	★★	–	★★	★★	★★★	★
HPCToolkit	★★★	★★	★★	★★	★★	★★	★	★★	★	★★	★★
IPM	★★★	★★	★★	★★	★★	★★	–	★★	★★	★★★	★
LIKWID	★★	★★★	–	–	–	–	–	★★	★★	★★	★★
MAQAO	★★★	★★	★★	★★	★★★	★★	★★	★★★	★★★	★★	★
memP	–	–	★★★	–	–	–	–	★★	★★	★★	★
MPE	–	–	–	–	★★★	–	★★	★★	★	★★	★★
mpiP	–	–	–	–	★★★	★★	–	★★	★★	★★★	★★
MUST	–	–	–	–	★★	–	★★	★★	★★	★★	★★

Untersuchte allgemeine Tools

	General	HW Counters	Memory	File System	MPI	Sampling	Tracing	Ease of Use	Automatability	Overhead	Availability
ompP	⋆⋆	⋆	–	–	–	–	⋆	⋆	⋆⋆	⋆⋆	⋆
OProfile	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆	–	⋆⋆⋆	⋆⋆	⋆⋆	⋆⋆
O|SS	⋆⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆⋆	⋆⋆	⋆
PAPI	–	⋆⋆⋆	–	–	–	–	–	⋆⋆	⋆⋆⋆	⋆⋆	⋆⋆⋆
Paradyn	–	–	–	–	–	–	⋆⋆⋆	⋆	⋆⋆	⋆⋆	⋆⋆
PMPI	–	–	–	–	⋆⋆⋆	–	⋆⋆	⋆⋆	⋆⋆	⋆⋆⋆	⋆⋆⋆
Score-P	⋆⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆	–	⋆⋆⋆	⋆	⋆⋆⋆	⋆⋆	⋆⋆
Slurm	⋆⋆	–	⋆⋆	⋆⋆	⋆⋆	⋆⋆	–	⋆⋆⋆	⋆⋆⋆	⋆⋆	⋆⋆⋆
TAU	⋆⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆⋆	⋆⋆	⋆⋆
Valgrind	–	–	⋆⋆⋆	–	–	–	⋆⋆	⋆⋆	⋆⋆	⋆	⋆⋆

Inhalt

Monitoring- und Profiling-Tools

1. Monitoring
   - GNU Time (Überblick über Prozessmetriken),
   - procfs
   - SYSSTAT (Systemmonitoring am Beispiel sar),
2. Profiling
   - Intel MPI (Statistics Gathering Mode),
   - ScoreP

procfs

1. *procfs* ist ein virtuelles Dateisystem, welches als Schnittstelle zum Kernel dient, um Prozess- oder Systeminformationen auszulesen oder zu manipulieren
2. In Linux besitzt jeder Prozess einen eigenen Ordner /proc/pid (pid = PID des Prozesses)
3. In der Regel sind die Dateien in diesem Ordner leer (Dateigröße gleich Null) und nur auf Anforderung werden diese durch den Kernel mit den entsprechenden Informationen gefüllt
4. Monitoring, knotenbasiert, Zeitreihe, als auch Übersicht (mit entsprechenden Programm auszulesen), Ergebnisse pro Prozess
5. Overhead: Kaum (max. 2 Prozent), unabhängig vom Messintervall
6. Sehr reichhaltige Quelle an Informationen

GNU Time

1. In seiner „geschwätzigen“ Variante (/usr/bin/time -v -p) liefert GNU Time eine über die Laufzeit summierte oder gemittelte Darstellung bestimmter Metrikwerte
2. Dazu gehören:
   - CPU: Verbrauchte CPU-Zeit (wall-clock time), vom Job/Prozess genutzte CPU-Zeit (in Prozent)
   - CPU: Zeit, die die CPU/Kern im User- oder Kernelmode verbracht hat (User-, Systemzeit in Sekunden)
   - Speicher: Hochwassermarke des Prozesses (RAM)
   - Swap/Paging: Anzahl der geringfügigen und bedeutenden Seitenfehler (minor/major page faults)
   - I/O: Anzahl an Dateisystemzugriffen (Input, Output zusammengefasst)
3. Monitoring, knotenbasiert, Ergebnisse prozessorientiert, Zusammenfassung der Ergebnisse (keine Zeitreihe), kein Overhead

sar

1. **sar** ist ein Programm aus den *SYSSTAT utilities*
2. Protokolliert knotenweise an bestimmten Zeitstempeln (z.B. nach jeweils 5 Sekunden) für eine bestimmte Anzahl an Messungen bestimmte Parameter
3. Monitoring, knotenbasiert, Zeitreihe, Ergebnisse nur knotenbasiert
   - CPU: sowohl über alle Kerne gemittelte, als auch Werte pro Kern,
   - CPU: Zeit, die die CPU im User-, Systemmode oder Idle verbracht hat (User-, System-, Idlezeit),
   - Speicher: Freier Speicher in KBytes und Prozent
   - Swap: Freier und benuzter Swapspeicher
   - Paging: Speicher, welcher pro Sekunde gepaged wurde (in Kbytes)
   - System: CPU-Auslastung
   - I/O: Anzahl an Daten, welche pro Sekunde geladen/gespeichert wurden (in Blocks/s)
   - Netzwerk: Summe, der pro Sekunde gesendeten und ermittelten Pakete oder Daten.

Intel MPI - Statistics Gathering Mode

1. Intel MPI ermöglicht die Sammlung von MPI-Kommunikationsinformationen und MPI-Kommunikationsmustern
2. Wird aktiviert durch Setzung der Umgebungsvariable I_MPI_STATS
3. Knotenübergreifend, geringer Overhead (max. 0.2 %), Ergebnisse gesamt, aber auch pro Prozess, Kommunikationsart (Collective vs. Point to Point), etc.
4. Metriken
   - Übertragendenes Datenvolumen im Job (aufgeschlüsselt nach Prozessen, als auch nach Collectives vs. P2P)
   - Anzahl der Aufrufe ausgewählter MPI-Calls,
   - Min/Max/Avr/Gesamtransferzeit in Millisekunden pro MPI-Call

Score-P

1. Toolsuite, welche zum Profiling, Event Tracing und zur Analyse eingesetzt werden kann
2. Folgende Schritte müssen durchgeführt werden:
   - Vorbereitung der Messung durch Instrumentierung (z.B. Compiler-Instrumentierung)
   - Messung (Konfiguration ob Event Tracing durchgeführt oder Call Graph erzeugt werden soll)
   - Analyse/Visualisierung der Ergebnisse z.B. durch CUBE oder Vampir (letzteres falls Tracing)
3. Betrachtung des textuellen Flat Profils (ähnlich zu gprof) ist oft schon hilfreich
4. Unter anderem werden Anteil MPI und Anwendungen dargestellt, dito Auflistung MPI-Calls (Häufigkeit, Verweildauer)
5. Profiling, knotenübergreifend, Zusammenfassung oder Zeitreihe

Inhalt

Batchsystem (hier Slurm)

1. Slurm ohne das „hdf5-Plugin“ liefert mit dem Kommando *sacct* folgende Werte:
   - Submit-, Start- und Endzeit, AllocCPUs, AllocNodes, NNCPUS, NNODES
2. Slurm mit dem „hdf5-Plugin“ liefert mit dem Kommando *sacct* dann zusätzlich:
   - AveCPU, AveCPUFreq,
   - AveDiskRead, AveDiskWrite, MaxDiskRead, MaxDiskWrite,
   - AvePages, MaxPages,
   - AveRSS, MaxRSS, MaxRSSNode, MaxRSSTask,
   - MaxDiskReadNode/Task, MaxDiskWriteNode/Task,
   - MaxPagesNode, MaxPagesTask,
   - MaxVMSize, MaxVMSizeNode, MaxVMSizeTask,
   - SystemCPU, UserCPU, TotalCPU,

   wobei manche nicht gefüllt werden.

HPCToolkit

1. HPCToolkit ist ein Toolkit, um die Programmperformance zu messen und diese zu analysieren.
2. Nutzt statistisches Sampling auf Basis von Timern und Hardwarecountern, was zu geringen Overheadzeiten führt (1-5 %)
3. Wichtige Eigenschaften/Ausgaben:
   - CPU: Wall-clock time, real time und CPU time
   - Speicher: Speicherallokierung, -deallokierung und Speicherlecks werden ermittelt
   - I/O: Gelesene/geschriebene Bytes auf Disk
   - MPI, OpenMP: Metriken werden für jeden Prozess und jeden Thread ermittelt und auf diesen können Operationen wie Summe/Max/Min/ über alle Prozesse/Threads durchgeführt werden

IPM

1. IPM (Integrated Performance Monitoring) ist eine portable Profilinginfrastruktur für parallel Programme, welche skalierbar ist, einen sehr geringen Overhead besitzt und keine Codemodifikation benötigt.
   - Peformancemetriken: Flops/s und weitere (ermittelt via PAPI),
   - Speicher: High Water Mark; Nutzung von getrusage, um weitere Daten zu bekommen (siehe auch GNU time)
   - OpenMP: Details pro Thread bzgl. Lastverteilung, um mögliche Ungleichgewichte zu ermitteln
   - MPI: Kommunikationstopologie und Statistiken für jeden MPI-Aufruf
   - I/O: Lesen von / Schreiben auf Disk

MAQAO

1. MAQAO (Modular Assembly Quality Analyzer and Optimizer) ist eine Toolsuite, welche Performanceanalyse und Optimisierung bietet und auf der Binärcodeebene arbeitet.
2. Bietet statistische Analyse und Instrumentierung (mittels MAQAO lightweight profiler (LProf))
3. Erlaubt unter anderem Aufgliederung der Ergebnisse, in welchen Bereichen wieviel Zeit verbraucht wurde, z.B. Anwendung, MPI, OpenMP, I/O, System (in Prozent).
4. Schleifen- und Funktionshotspots werden (hoffentlich) erkannt und können über einen Report angezeigt werden (werden durch LProf ermittelt)
5. Weiterhin werden Reports erstellt (einfach oder fortgeschritten), in denen Optimierungspotentiale erläutert werden und wie man die Bottlenecks beseitigen kann (Code Quality Analyzer)

mpiP

1. mpiP ist eine leichtgewichtige und skalierbare Profiling-Bibliothek für MPI-Anwendungen.
2. Erfasst für jede MPI-Funktion durch statische Informationen (Sampling) bestimmte Metriken
3. Geringer Overhead aufgrund von Sampling, anstatt Tracing
4. Alle Informationen sind Task-local und werden am Ende des Jobs zusammengeführt.
5. Bericht liefert u.a. folgende Informationen:
   - Grundlegende Werte, wie Start- und Endzeit, PID jedes Tasks
   - Schnellübersicht über die 20 meisten MPI-Calls, welche aggregiert die meiste Zeit in dem Job verursachen
   - Analog für die 20 häufigsten MPI-Calls, welche aggregiert die meisten Daten gesandt haben
   - MPI I/O report

ompP

1. Profiling-Tool für OpenMP
2. Ermittlung der Werte von Hardware Performance Counters unter Benutzung von PAPI
3. Darstellung als Call Graph oder Flat Profile (ähnlich zu gprof)
4. Overhead-Report (z.B. bzgl. Lastverteilung) und Auflistung von Ineffizienzregionen

Weitere Tools/Interfaces

1. Paradyn
2. PMPI
3. Open|SpeedShop
4. TAU (Tuning and Analysis Utilities)
5. vmstat
6. Für weitere siehe
   `https://profit-hpc.de/wp-content/uploads/2017/04/profithpc_deliverable_2-1_final.pdf`