Universität Hamburg
Department Informatik
Wissenschaftliches Rechnen
Prof. T. Ludwig, J. Kunkel, N. Hübbe

Übungsblatt 4 zum Praktikum
Paralleles Programmieren
im SoSe 2013

1 Leistungsanalyse

1.1 VampirTrace & Vampir

Mit Hilfe von VampirTrace können die Aktivitäten eines Programmes in sogenannten Spurdaten aufgezeichnet und später analysiert werden. Einerseits können OpenMP und MPI Aufrufe als auch Funktionsaufrufe Ihres Programmes erfasst werden. Um VampirTrace nutzen zu können müssen Sie Ihr Programm neu Compilieren. Anstelle von `gcc` oder einem anderen Compiler verwenden Sie: `vtcc -vt:inst compinst -vt:cc mpicc`
Hierbei werden alle Funktionen Ihres Programmes instrumentiert und können somit analysiert werden. Vorsicht, dies kann bei kleineren Funktionen den Overhead drastisch erhöhen. Anstelle von `compinst` können Sie auch `manual` verwenden, in dem Fall wird nur MPI instrumentiert.
Bei der Ausführung ihres Programms mittels `mpiexec` werden Spurdaten mit der Erweiterung otf für den Programmlauf erstellt, welche Sie mittels `vampir` visualisieren können. Dabei muss `vampir` direkt auf unserem cluster ausgeführt werden, wozu es notwendig ist, dass für die SSH-Verbindung X-Forwarding aktiviert ist. Unter Linux muss dazu einfach das Flag -X beim SSH-Aufruf angegeben werden ßsh -X mustermann@cluster.wr.informatik.uni hamburg.de".
VampirTrace kennt viele Einstellungsmöglichkeiten über Umgebungsvariablen. Dies kann dazu genutzt werden um ein Profil für den Anwendungslauf zu erstellen. Dafür müssen Sie die Umgebungsvariable `VT_MODE` vor der Programmausführung auf STAT setzen. Beispielsweise in der Bash: `export VT_MODE=STAT`
Es ist ebenso möglich ein Profil aus einem Trace zu erstellen: `otfprofile -i a.otf .`
Mehr Information finden Sie in kompakter Form auf dem CheatSheet: `http://www.vampir.eu/public/files/pdf/vtcheatsheet_a4.pdf`

1.2 Likwid

Likwid ist ein sehr leichtgewichtiges Werkzeug zur Analyse von verfügbaren Hardware-Countern der CPU. Somit lassen sich bspw. Flop/s oder Speicherdurchsatz der Anwendung ermitteln um Leistungsengpässe in der Programmierung aufzudecken.
Um Likwid nutzen zu können müssen Sie das entsprechende Modul laden.

```
$ module load likwid
```

In einem Batch-Skript wird Likwid so aufgerufen, dass es ein weiteres Programm mit den angegebenen Parametern aufruft.

```
likwid-perfctr -C 1-12 -g <GRUPPE> <PROGRAMM> <PARAMETER>
```

Hardwarebedingt kann bei einem Aufruf immer nur eine Gruppe gemessen werden, beginnen Sie typischerweise mit der Gruppe VIEW.
Auf unserem System gibt es folgende Gruppen:

```
BRANCH: Branch prediction miss rate/ratio
CACHE: Data cache miss rate/ratio
DATA: Load to store ratio
FLOPS_DP: Double Precision MFlops/s
FLOPS_SP: Single Precision MFlops/s
FLOPS_X87: X87 MFlops/s
L2: L2 cache bandwidth in MBytes/s
L2CACHE: L2 cache miss rate/ratio
L3: L3 cache bandwidth in MBytes/s
L3CACHE: L3 cache miss rate/ratio
```

`MEM: Main memory bandwidth`
`TLB: TLB miss rate/ratio`
`VIEW: Overview of arithmetic and memory performance`

Weitere Information finden Sie unter: `http://code.google.com/p/likwid/`

2 Praktikumsprojekt

Zum erfolgreichen Abschluß des Praktikums entwickeln Sie im Team eigenständig eine parallele Anwendung. Im Folgenden ein paar Hinweise zum Aufbau der von Ihnen zu erstellenden Artefakte (Kurzbeschreibung, Präsentation der Aufgabenstellung, Programmcode, evtl. Ergebnissvideo, Abschlusspräsentation & Ausarbeitung).
Zunächst stellen Sie in einer kurzen Präsentation Ihre gewählte Aufgabe vor, **abzugeben ist ein PDF**:

1. Kurzbeschreibung des Problems
2. Lösungsansatz (sequentielles Programm)
3. Flexibles Parallelisierungsschema (wie teilen Sie das Problem auf die Prozesse auf)

Bei der Programmierung und Auswertung Ihrer Lösung gehen Sie wie folgt vor, **abzugeben ist der Sourcecode als tar.gz & evtl. ein Video mit Ergebnissen (bitte komprimiert)**:

1. Implementation eines sequentielles Programms (oder Sie passen ein bestehendes Programm an)
2. Leistungsanalyse des sequentiellen Programms mit `gprof` und `likwid`. Erreicht Ihr Programm die von Ihnen erwartete Leistung? Evtl. kleine Optimierungen vornehmen.
3. Parallelisierung des Programms mit MPI
4. Optional: Hybride Parallelisierung des MPI-Programms mit OpenMP
5. Erstellung eines Profils und/oder Spurdatenanalyse des Programms. Verbringt Ihr Programm die meiste Zeit in der Bereichnung der Lösung? Optimieren Sie evtl. das Kommunikationsmuster oder erhöhen Sie die Komplexität des sequentiellen Programms.
6. Skalierbarkeit – Erstellen Sie ein Speedup-Diagramm für eine feste Aufgabenstellung

Abschlußpräsentation und Ausarbeitung mit folgendem Inhalt, **abzugeben sind zwei PDFs**:

1. Problemstellung
2. Lösungsansatz
3. Parallelisierungsschema
4. Laufzeitmessungen
5. Leistungsanalyse
6. Skalierbarkeit
7. Evtl. Grafiken der Programmausgabe

Für die Veröffentlichung Ihrer Ergebnisse auf unserer Webseite benötigen wir außerdem eine Kurzbeschreibung, **abzugeben ist ein Textdokument (.txt oder ähnliches, *kein Word*!)**.