A scalable method for identifying and displaying the communication behaviour of large scale applications

Christoph Geile

Juelich Supercomputing Centre, Research Centre Juelich

January 2008
1 Motivation

2 Communication analysis with SALSA

3 Results visualization in R

4 Usage examples

A scalable method for identifying and displaying the communication behaviour of large scale applications
Motivation

Communication analysis with SALSA

Results visualization in R

Usage examples

Usage of parallel systems

- expensive simulations and calculations in science
- divide calculation to many processors
  - parallel processing
  - interprocess-communication
- compute time is expensive
  \[\Rightarrow\] reduce this time
Event Tracing

Predefined events are determined and stored while running the application. Examples are
- entering or leaving a function
- sending or receiving a message
- ...

⇒ trace file contains all important events, including their time-stamp, location, type, etc.
State of the art: tracing tools

- manual analysis by visualizing
- scales only up to a few hundred processors

A scalable method for identifying and displaying the communication behaviour of large scale applications
State of the art: tracing tools

- scalable automatic search for performance problems
- no detailed communication analysis

⇒ scalable detailed communication analysis necessary
SALSA

Statistical Analysis of Large Scale Applications

1. parallel creation of communication statistics
   - calculation
   - storage

2. visualization in form of a colored matrix

Even for a large number of processors
Performance metrics

The following performance metrics can be calculated for every pair of processors:

- count
- length
- duration
- rate
- min
- max
- avg
- sum
Calculation of the performance metrics

1. read next requested performance metric
2. read necessary events and their attributes from trace files
3. **Every** processor creates an array (length = number of processors), which stores the calculated values
4. create and store the complete matrix
5. calculate more performance metrics?
   back to (1)
Calculation of the performance metric 'rate'

- identify sending process
- read number of bytes out of send-event
- calculate difference of time stamps
- ratio of bytes and time
- determine min, max or avg
- set array field
Creating the communication matrix

- divide involved processors into blocks
- all values of one block are gathered at one processor
- values are written in block file
Types of storing the data

1. full matrices
   - blockwise output in form of matrix
   - algorithm to convert array is necessary

2. sparse matrices
   - blockwise compression of matrix
   - diagonal run through matrix parallel to main diagonal
   - store a value and how often it appears in series
Sparse matrices

order of diagonal cycles (example values in matrix)

fields numbering

storing of matrix (value, quantity)

A scalable method for identifying and displaying the communication behaviour of large scale applications

Christoph Geile

Juelich Supercomputing Centre, Research Centre Juelich
## Result

Each performance metric is stored in a separate directory, which contains the statistical values as a blockwise matrix.
The statistical environment R

- implementation of statistical language S
- consists of language and environment
- free software
- offers a huge number of functions for statistical analysis of data
- extensible

A scalable method for identifying and displaying the communication behaviour of large scale applications
Visualization of performance metrics

- for every matrix block:
  - read data blockwise
    ⇒ limited storage place
  - reduction of matrix dimensions to displayable size
    ⇒ blockwise compression by generating
      - minimum
      - maximum
      - average
      - sum
  - display data
Visualization options of the SALSA-matrix

1. select a region
2. fixed and adapted colors in region
3. elimination of outliers

A scalable method for identifying and displaying the communication behaviour of large scale applications
Select region and coloring

- select region at function call
- adapt colors to region or to complete matrix
Elimination of outliers

outlier = single value which doesn’t fit to the rest

- percentage can be cut off by generating quantile
- coloring is adapted to remaining values
- region is narrowed down
Return matrix for further processings

In addition to showed visualization the matrix of course can be processed as R-object. Possibilities are manyfold, e.g.

- apply statistical calculations
- combine different performance metrics
- further graphical illustrations in R (e.g. boxplot, barplot)
Usage examples

1. Communication behaviour at increasing number of processors (smg2000)
2. Options to reduce dimensions
3. Advantage of sparse matrix method
4. System analysis of BlueGene/P “JuGene“
   - 16384 4-way SMP processors => 65536 processors
   - Processor type: 32-bit PowerPC 450 core 850 MHz
   - Linpack: 167 Teraflops
   - 2nd in Top500-list November 2007
Communication behaviour at increasing number of processors

⇒ no rearrangement of common behaviour

Christoph Geile
Juelich Supercomputing Centre, Research Centre Juelich

A scalable method for identifying and displaying the communication behaviour of large scale applications
options of reduce dimensions

minimum

maximum

average

sum
Storage need for sparse matrix

improvement of 41 to 88 %
improvement of 85 to 97 %
System analysis of BlueGene/P “JuGene“

- display of system tool “linktest“ not scalable
- solution: visualizing of measured data with SALSA
- result: particular irregularities identified

communication of test application with 8192 processors

region of a matrix with 32768 processors
Summary

- developed parallel scalable calculation of communication statistics
- developed scalable visualization of communication data
- reduced storage requirement by 97%
- versatile technique
Thank you for your attention!
PEARL

- Schnittstelle zu den Trace-Dateien innerhalb des Scalasca-Projektes
- Parallele Ausführung
- Erlaubt das Erkennen von Ereignissen
  ⇒ Zugriff auf Sende- und Empfangsereignisse mit genauen Informationen über Zeit, Nachrichtenlänge, beteiligte Prozessoren, etc.
- Ermöglicht das Versenden kompletter Ereignisse mit allen enthaltenen Informationen
Der CallbackManager

- Jede Trace-Datei wird einzeln durchlaufen
- Anwender kann bestimmtes Verhalten beim Auffinden registrierter Ereignisse vorgeben
- CallbackManager sucht nach Ereignissen und führt Vorgaben des Anwenders aus
Berechnung der Leistungsmetrikten

- Sendeprozess ermitteln
- Differenz der Zeitstempel berechnen
- min, max, avg oder sum bestimmen
- Vektorfeld setzen

MPI_Send

MPI_Recv

location

Duration

time

Christoph Geile
Juelich Supercomputing Centre, Research Centre Juelich

A scalable method for identifying and displaying the communication behaviour of large scale applications
Berechnung der Leistungsmetriken

- Sendeprozess ermitteln
- Anzahl der Bytes aus Sendereignis auslesen
- min, max, avg oder sum bestimmen
- Vektorfeld setzen
Berechnung der Leistungsmetriken

location

Count

MPI_Send

MPI_Recv

- Sendeprozess ermitteln
- Vektorfeld inkrementieren
Einlesen der Daten

**read.table()**

- Vollbesetzte Matrizen können blockweise eingelesen werden
  ⇒ Nur begrenzter Speicherplatz
- Dünnbesetzte werden zunächst in vollbesetzte umgewandelt
  - Schleifen in R sehr ineffektiv
  ⇒ Funktion `pipe()` zur Wiederherstellung durch ein C-Programm

```
1 1
0 5
1 4
0 5
1 1
1 0 0 1
0 1 0 0
0 0 1 0
1 0 0 1
```
Reduktion der Matrixdimensionen

- Je nach Ausgangs- und Zieldimension werden Felder zusammengefasst
- Mögliche Optionen dafür sind Minimum, Maximum, Durchschnitt und Summe
Rechenzeit zur Erstellung und Speicherung

- Kurven nähern sich an
- Akzeptabler Zeitbedarf für höhere Prozessorzahlen wahrscheinlich

A scalable method for identifying and displaying the communication behaviour of large scale applications
Zeitbedarf zum Erstellen der Präsentation in R

A scalable method for identifying and displaying the communication behaviour of large scale applications