Increased data distribution awareness within a compiler

Sarah Webster
CSE @ UNSW

<sarahw@cse.unsw.edu.au>

November 2006

Background

System: Commodity Clusters

Program: Data Parallel

Data: Irregular

Background

System: Commodity Clusters

Program: Data Parallel

Data: Irregular

Aim: improve performance using load balancing
(info + decide + execute)

- balanced data - equally distributed
- unbalanced data - not balanced

Problem

- regular load balancing solutions are not applicable
- irregular data solutions are very specific
- RTS performs all load balancing steps

Compiler Awareness

Question: How useful is compiler awareness of more advanced data distributions in enabling more effective load balancing?

Advantages:

- Greater flexibility in load balancing decisions
- Compiler can outline more choices for runtime system
- More specific in categorising data distributions
- No need to balance simply to find out more information

New Data Distributions and Characteristics

Extend the data distributions:

- BalData - equally distributed data
- BalSeg - BalData, but without split segments
- KImb - imbalanced, in a globally known way
- Imb - unknown distribution
- OutOfOrder - data order is explicit

Data Characteristics:

Useful for fine tuning load balancing decisions

- Degree of imbalance: difference in amount of data on each node
- Degree of irregularity: difference in size of data segments
- Number of segments
- Average segment size

EXAMPLE: PERMUTE

inputs: data array and index array

outputs: re-arranged data array

before: balance both arrays, then run function

EXAMPLE: PERMUTE

inputs: data array and index array

outputs: re-arranged data array

before: balance both arrays, then run function

after:

- compare distributions, analyse data characteristics
- re-distribute data from one/both arrays as needed
- run function

LIMITATIONS

- distributions a good guide, but can be ambiguous
- difficult to compare two/more distributions
- difficult to optimise load balancing code
- greater flexibility could lead to large overhead

Results

BalData vs BalSeg:

Function: segmentSum

Data: segments that are split across node boundaries unless placed out of order

Benchmark: BalData vs BalSeg data + unordered

Result: 10000 data elems on 8 node cluster: balSeg is 2 orders magnitude faster