Real-time Grid Computing: Recent Results on a Pilot Job Approach

Hermann Heßling

7. 2. 2011
DESY Computing Seminar
University of Hamburg
Outline

1 Grid Computing: Real-time Aspects
2 Chess on the Grid: GriScha
   2.1 Chess Computing
   2.2 GriScha
3 Peer-to-Peer Concepts in Real-time Grid Computing
   3.1 Messages
   3.2 Shared Memory
4 Summary and Outlook
1. Multiple Processor Systems

Communication models

a) A shared-memory multiprocessor
b) A message-passing multicomputer
c) A wide area distributed system
Nonuniform Memory Access (NUMA)
- Single address space visible to all CPUs
- Access to remote memory is slower than to the local memory

A Distributed Shared Memory (DSM) architecture needs a lot of internal messages
1. Real-time Computing

- Some characteristics
  - Computing system reacts within constraints on the response time
  - Deadlines must be fulfilled independently of the load of the system
  - Hard real-time systems: no latency tolerated (for example chess)
  - Soft real-time systems: some latency tolerated (for example chess)

- Real-time operating systems
  - Framework for real-time applications
  - Advanced scheduling algorithms (priority ~, static time ~, ...)
  - Interrupt handler
  - Fast memory allocation algorithms
  - Examples: QNX, RTLinux, ...
1. Grid Computing

- The broker selects a gateway (GW) which passes the user job to a worker node (WN).
- The WNs within a grid site are protected against the outside by a firewall.
- WNs cannot react directly on real-time user events.
What is *real-time grid computing*?

- Colleagues in the spring of 2009: “never heard about it.”
1. Real-time Grid Computing

- Google (Oct. 24, 2010): 63 hits
Google (Febr. 3, 2011): 56,100 hits
Google (Febr. 3, 2011): 66 hits $\Rightarrow$ + $\sim$1 hit / month
Google (Febr. 3, 2011): 56,100 hits

1. Real Time Monitor
   The Real Time Monitor. Welcome to the home page for the Real Time Monitor (RTM). RTM is a visualisation of activity of the grid computing infrastructure... rtm.hep.ph.ic.ac.uk/ - Im Cache

2. PoS(GiF2006)???
   Dateformat: PDF/Adobe Acrobat - Schnellansicht
   von R di Meo
   4 Feb 2006 ... Real-time grid computing for financial applications. Stefano Cozzini... INFN Democritos and EGRID project. E-mail: cozzini@democritos.it... pos.sissa.it/archive/conferences/026/007/GRID2006_007.pdf

3. Euro-Par 2004 parallel processing: 10th International Euro-Par ... - Google Ergebnisseite
   Marco Danelutto, Domenico Laforenza, Marco Vanneschi - 2004 - Computers - 1081 Se 1 Introduction Research in real-time Grid computing is needed to enable Grid service newly emerging class of large-scale real-time distributed... books.google.de/books?isbn=3540229248...

4. Grouping-Based Job Scheduling Model in Grid Computing
   Dateformat: PDF/Adobe Acrobat - Schnellansicht
   provides a real time grid computing environment and reduces the waiting time of the grouped jobs. In future, this work can be extended to design a parallel...
1. Real-time Grid Computing

- Google’s No. 3
- M. Eltayeb, A. Dogan, F. Ö zgüner: A Path Selection Based Algorithm for Maximizing Self-satisfiability of Requests in Real-Time Grid Applications (Euro-Par 2004)

First sentence of the introduction:

“Research in real-time Grid computing is needed to enable Grid services for newly emerging class of large-scale real-time distributed applications.”

Goals: find a scheduling algorithm that

- distributes efficiently huge datasets between grid nodes connected over by a wide area network
- accounts for deadlines by which datasets have to be delivered

Dijkstra’s routing algorithm is extended by a priority component

- If different routing paths cross the same intermediate node, the access of low-priority paths to this node may be blocked by a certain amount of time
1. Real-time Grid Computing

- **Google’s No. 2**: S. Cozzini, R. di Meo, E. Corso: *Real-time grid computing for financial applications* (GRID2006)
  - **Goals**
    - Porting a financial application on a grid infrastructure (EGRID)
    - Guarantee real time computing
  - **Master/slave architecture**
    - Master: running outside the grid
    - Slaves: grid worker nodes, allocated in advance
    - Limitations: poor scaling due to network overhead
  - **Modification: continent algorithm**
    - Master and grid site communicate via a “bridge node” over the public internet
    - Worker nodes within a grid site communicate over local network
  - **Performance tests**: 1-20 nodes
Google’s No. 1

2006: “Real-time grid computing monitor maps” of the worldwide activity of the LHC Grid (LCG)
ChessBrain Project (www.chessbrain.net)

- Jan. 30, 2004: ChessBrain was running on 2070 computers connected over the internet
  - Draw against GM Peter Heine Nielsen (2683 ELO rating)
- Guinness World Record for “the largest network chess computer”
- Communication SuperNode ↔ PeerNodes: SOAP (HTTPS+XML)
  - Overhead ~ 10 sec. per move

ChessBrain II

- Main goal: increase scalability
- Hierarchically distributed communication model (SuperNode, ClusterNodes, PeerNodes)
- SOAP replaced by ~SIP (⇒ VOIP)
- Parallel game tree search (APHID)
- Status: “server testing has begun” (2006)
Outline

1. Grid Computing: Real-time Aspects

2. Chess on the Grid: GriScha
   2.1 Chess Computing
   2.2 GriScha

3. Peer-to-Peer Concepts in Real-time Grid Computing
   3.1 Messages
   3.2 Shared Memory

4. Summary and Outlook
2.1 Chess Computing

- Some estimates
  - Ken Thompson (Unix, B, C): 1 Ply ~ 200 ELO (↔ Belle vs. Belle)
  - Rapid chess: 40 moves in 15 minutes ⇒ ~20 s/move
  - Assumptions
    - Computational power: ~ 1 million positions/s
    - Middlegame: ~ 35 legal moves
    - Parallelizing chess:
      - 1 worker node: 4.7 Ply
        - 20 * 10^6 ~ 35^{4.7}
      - 500 worker nodes: 6.5 Ply
        - 500 * 20 * 10^6 ~ 35^{6.5}
  - Horizon shift on 500 nodes: ≤ 1 move

Working Group

- Daniel Heim B. Sc. (2009 - )
- HH
- Christian Lehmann (2009)
- Christoph Neumann B. Sc. (2009 - )
- Christian Rossius B. Sc. (2009 - )
- Marco Strutz B. Sc. (2009 - )
- Francesco Tietke B. Sc. (2010)
2.2 GriScha

- Very simple chess engine
  - Elementary evaluation of chess positions
  - No external intelligence
    - No database of chess openings
    - No endgame tablebase
    - No database of chess games

- Can you make many beginners stronger than a master?
  - Grid computing provides access to thousands of worker nodes
  - The challenge is to provide collective communication in huge communities

- Goal
  - Developing a testbed for exploring real-time communication in grid computing
2.2 Grischka

- **Architecture**
  1. Move generator
     - Board representation
     - Creates a list of legal moves for a given chess position
  2. Evaluation function = (unweighted) sum of two contributions:
     - Value of pieces (pawn = 1, knight = bishop = 3, ...)
     - Position evaluation (“A knight on the rim is grim”)
  3. Search strategy
     - Alpha-Beta pruning
       - cutting the search tree without modifying the result
       - ~10 times faster than the minimax algorithm
     - Iterative deepening
     - Parallelization
  4. Communication between MasterNode and grid worker nodes
  5. Graphical user interface
2.2 Grischka

- **Value of the pieces** (opponent: values multiplied by -1)
  - pawn 10
  - knight 30
  - bishop 32
  - rook 55
  - queen 98
  - king 1000
  - The usual values are multiplied by 10 and modified by some oddness to reduce the probability that different positions obtain the same evaluation number. (The oddness is partly motivated by Larry Kaufman’s statistical analysis.)

- **Position evaluation** (opponent: values multiplied by -1)
  - two bishops + 5
  - “knight on the rim is grim” - 4
  - pawn at rank n (n<6) + 2n
  - pawn: isolated, doubled, behind - 3
  - castling still permitted + 5
  - attacked Q, R, B, N - 0.09*value
  - attacked K - 0.008*value
2.2 GriScha

- Communication
  - Gatekeeper
    - allocates worker nodes in advance
    - submits a pilot job to worker nodes
  - Worker node
    - runs a pilot job
    - firewalls of grid sites must allow at least 1 outbound connection
  - Pilot job
    - contains chess evaluation software (JAVA)
    - initiates a connection to the MasterNode
    - receives requests from MasterNode and answers in a given time
  - MasterNode (in conjunction with the GriScha Engine, see p. 24)
    - calculates the legal moves after the first 3-4 Ply
    - distributes chess positions to the worker nodes
    - evaluates results from the worker nodes and selects a move
2.2 GriScha

- Communication between MasterNode and worker node
  - Simple Invocation of Methods over Network (SIMON)
  - “SIMON is the RMI alternative” (http://dev.root1.de/wiki/2)
  - Client establishes **one** TCP/IP connection to the server
  - Client can call methods on the server, over the **same** connection server can call methods on the client (callback)
  - Scalable (⇔ Java New IO (NIO) and thread pools)

For comparison: RMI needs at least two connections to call methods on both sides.
  - Problem: the firewalls of grid sites block inbound communication
2.2 GriScha

- GriScha Engine – a collection of chess specific components
  - Connected via SIMON to the MasterNode
    - virtual worker nodes (VWN) are activated by the MasterNode
    - the Worker Node (WN) Manager acts as a proxy
  - Calls methods on the VWNs which are send via SIMON to the real worker nodes

- XBoard is a graphical chessboard (open source)
  - The GriScha Engine is connected to XBoard via the standard IO (STDIO)
Ping test

- The MasterNode calls periodically (1000 Hz) an empty method on the worker nodes and measures the duration until he receives a reply.
- Result: the latency is independent of the number of worker nodes.
  - The band structure is due to the quality of the internet connection between the MasterNode and the grid sites.
2.2 SIMON in Real-time Grid Computing

Message exchange
- The MasterNode calls periodically (0.2 Hz, 12 periods) a method on each worker node which send back a “packet” of P bytes. Measured is the time between calling the method and receiving the complete message.

Results
- Latency grows with the message length
- An influence of the number of worker nodes is not observed

• Message event
  - Every minute the message size P is increased by 0.1 MB.
  - For reasons of readability a random number 0...0.09 MB is added to the message size P before an event is noted in the scatter plot

Number of active worker nodes at time \( t \)
- 2 worker nodes (of the total 80) did not start in time
2.2 Counter-productive Parallelism

- Clients are called periodically (1 Hz, 5 periods) by a server to send back to him a randomly generated string message of N bytes.
  - Communication between server and clients: SIMON over a switched Fast Ethernet.
- Results of speedup measurements (● ● ●)
  - Standard deviation of $T(1) \otimes T(p)^{-1} \leq 6\%$

Let a program consist of a sequential and a parallelizable part:
- $T_{seq}$ = time to run the sequential part.
- $T_{par}$ = time to run the parallel part.

**Amdahl’s law**: time to finish the program on $p$ parallel processors

$$T(p) = T_{seq} + \frac{T_{par}}{p} + T_{overhead}$$

Assumption: $T_{overhead} = c(p - 1)$

Least-square fit of the Speedup = $\frac{T(1)}{T(p)}$ for long messages (green curve):

$$\text{Speedup} \approx \frac{1}{0.22 + \frac{0.78}{p} + 0.0070(p - 1)}$$

Results:
- Speedup is growing (~linearly if $p \geq 4$) for small messages (blue and red curve).
- $c \approx 0.007 \ T(1) > 0 \Rightarrow$ speedup decreases if a server receives too long messages from many clients at the same time.
Worker node topology
- ~500 worker nodes

Access to the grid VO DECH was kindly supported by
GriScha has an interface to Graphviz
- Graph visualization software (open source)

Network latency
- ~20 ms between Grid sites and MasterNode
- GriScha is equipped with a module for analyzing traffic
Outline

1. Grid Computing: Real-time Aspects
2. Chess on the Grid: GriScha
   2.1 Chess Computing
   2.2 GriScha
3. Peer-to-Peer Concepts in Real-time Grid Computing
   3.1 Messages
   3.2 Shared Memory
4. Summary and Outlook
3.1 JXTA

- 2001: JXTA initiated by Sun Microsystems
  - Standardization of peer-to-peer (P2P) applications based on Java
    - P2P
      - devices can act as client and as server
      - no central authority $\Rightarrow$ information distributed over P2P network
  - JXTA: “juxtapose” – positioned side-by-side
    - P2P model $\leftrightarrow$ client-server model
  - Set of open XML-based protocols
  - https://jxta.dev.java.net

- 2009: JXSE
  - JXTA Standard Edition
  - Open Source Java implementation of the JXTA protocols
  - http://jxse.kenai.com
3.1 JXTA

- JXTA acts as a standard to form a P2P network
  - **Routing** on application layer
  - **Discovery**: if no direct route to a destination is known, a Route Resolver Query is sent to the members of the peer group

- Accessing peers behind a firewall (+NAT) is not trivial
  - JXTA operates on TCP (or UDP) / IP
  - A JXTA peer is listening to (at least) one port for incoming messages
  - A JXTA peer is sending messages to a destination over specified ports
3.1 JXTA

- JXTA Transportation
  - Transportation concept: edge peers use pipes to send messages
  - Messages (XML/binary) may (or may not) reach the destination
  - The time needed to exchange a message between edge peers is not constrained
    - This point is of importance for real-time computing
  - Rendezvous peers: connect edge peers
    - Delegate queries to other rendezvous peers

- JXSE Transportation
  - Relay peers: connect (rendezvous) peers separated by firewalls/NAT
    - Enabled with a public IP address
  - Default TCP communication ports: 9701-9799
    - If these ports are blocked by a firewall, JXSE tries to connect to a known relay peer using HTTP(+XML) via the TCP port 9901
      - If port 9901 is also blocked internal peers are not accessible from WAN
3.1 JXTA in Grid Computing

Gatekeeper allocates worker nodes (WN) in advance by submitting pilot jobs to the broker.
The pilot jobs use **SIMON** to establish a connection between worker nodes and the MasterNode.
(At least) one peer per site becomes a rendezvous peer (RP).
MasterNode = relay peer? Advisable in real-time computing?
3.2 JuxMem

- JuxMem – Juxtaposed Memory
  - A Grid Data-Sharing Service
    - Based on JXTA
    - Management of shared memory data (not only files)
    - 2008 – JuxMem-C 0.5 released ("complete rewrite from scratch")
  - Unification of DSM and P2P technologies
    - Distributed Shared Management (DSM) systems
      - Platform: clusters
      - Scaling: small (~ 10²)
      - Architecture: static
      - Data consistency: important
    - Peer-to-Peer (P2P) systems
      - Platform: internet
      - Scaling: huge (~ 10⁶)
      - Architecture: dynamic
      - Data consistency: ---
JuxMem: a Grid Data-Sharing Service

- External grid data-sharing service
  - Grid-scale: $10^3$-$10^4$ nodes
  - Data persistence
  - Transparent data localization
  - Data consistency
  - Fault-tolerance

- JuxMem = DSM + P2P

Implementation
- Multiple replication strategies
- Configurable consistency protocols
- Based on JXTA 2.0 (http://www.jxta.org/)

Current collaborations
- French ACI MD: GDS and GdX projects
- Sun Microsystems, Santa Clara, USA
- Univ. New Hampshire, NH, USA
- AIST, Tsukuba, Japan

http://www.irisa.fr/paris/Juxmem

http://juxmem.gforge.inria.fr
Peers

- creating peers
  - `juxmem *peer = new juxmem(id, juxmem_group, listening_port);
    id = manager, client`

- manager peer
  - unique, handles communication between peers

- provider peers
  - publish size bytes in JuxMem network
    - `peer->share(size)`
- **client peers: writer**
  - allocate shared data within JuxMem network of size bytes
    - `void* ptr = peer->malloc(data_id, size,...)`
  - acquire/release a write lock on a JuxMem data
    - `peer->acquire(data_id)`
    - `peer->release(data_id)`
  - Example: (writing ‘a’ at position 4) `memset(ptr + 4, ‘a’, 1)`

- **client peers: reader**
  - map shared data within JuxMem of size bytes to a local pointer
    - `void* ptr = peer->mmap(data_id, size...)`
  - acquire/release a read lock on a JuxMem data
    - `peer->acquire_read(data_id)`
    - `peer->release(data_id)`
4. Summary and Outlook

- **GriScha**
  - A soft real-time grid computing system
  - Communication between MasterNode and grid worker nodes
    - Pilot jobs
    - SIMON protocol
  - Shown to run successfully on a grid system (gLite) of medium size
    - HTW Berlin (June 6, 2010): ~500 nodes, ~10 hours
    - DESY Hamburg (Febr. 7, 2011): ~650 nodes, ~2 hours

- **Current activities**
  - Exploring P2P methods for exchanging data between worker nodes
    - JXTA / JXSE ⇒ messages
    - JuxMem ⇒ shared memory

- **Future topic: approaching hard real-time grid computing**
  - Priority
  - Distributed semaphores