A case study on implementing virtual 5D torus networks using network components of lower dimensionality

HiPINEB 2017

Francisco José Andújar Muñoz et al.
Outline

Introduction

nDT torus topology

Building nDT torus using EXTOLL cards

Performance evaluation

Conclusions and future work
Outline

Introduction

nDT torus topology

Building nDT torus using EXTOLL cards

Performance evaluation

Conclusions and future work
Supercomputer systems

Many scientific problems cannot be addressed in a laboratory:

- Non-reproducible problems.
- Too dangerous experiments.
- Too expensive experiments.
- Different time constants for the systems and the experimenter.
- ...
Supercomputer systems

- Many scientific problems cannot be addressed in a laboratory:
  - Non-reproducible problems.
  - Too dangerous experiments.
  - Too expensive experiments.
  - Different time constants for the systems and the experimenter.
  - ...

- **Supercomputing is the key** to address these problems!

- Supercomputers are used in several scientific areas:
  - *Medicine*: Protein modelling involved in multiple diseases, as cancer, Alzheimer, etc.
  - *Meteorology*: Climate modelling and weather prediction.
  - *Physics*: Nuclear reactions, supernovae and black holes modelling, etc.
  - *Automotive industry*: Aerodynamics modelling.
Interconnection networks

- Supercomputers can have thousands of computing nodes.
- **The interconnection network is an essential component** to communicate this huge amount of nodes!
- The network topology has a significant impact on the overall system performance.
- Torus topology is widely used in supercomputers:
  - Constant radix $\Rightarrow$ Facilitates implementation.
  - Low Radix $\Rightarrow$ Simpler and cheaper hardware.
  - Scalable $\Rightarrow$ Linear cost of expansion.
  - Easy implementation of routing algorithms.
Building a n-dimensional torus

We can build a torus using:

- 2–port communication cards ⇒ 1D torus.
Building a n-dimensional torus

We can build a torus using:

- 2–port communication cards ⇒ 1\(D\) torus.
- 4–port communication cards ⇒ 2\(D\) torus.
Building a n-dimensional torus

We can build a torus using:

- 2–port communication cards ⇒ 1D torus.
- 4–port communication cards ⇒ 2D torus.
- 6–port communication cards ⇒ 3D torus.
Building a n-dimensional torus

We can build a torus using:

- 2–port communication cards ⇒ 1D torus.
- 4–port communication cards ⇒ 2D torus.
- 6–port communication cards ⇒ 3D torus.
- ...
- 2n–port communication cards ⇒ nD torus.
The torus performance depends on the number of dimensions.

- The higher number of dimensions:
  - The lower distances among nodes.
  - Consider a 1024 PE network:
    - $32 \times 32$ 2D torus: $d_{avg} = 16$
    - $16 \times 8 \times 8$ 3D torus: $d_{avg} = 8$
    - $4 \times 4 \times 4 \times 4 \times 4$ 5D torus: $d_{avg} = 5$
  - The higher network performance.
Increasing the torus performance

- The higher number of dimensions:
  - The higher number of ports.
  - The hardware complexity increases:
    - More expensive chip production.
    - Difficulty to implement some techniques (e.g. VOQ).
  - More expensive hardware.

- Is it possible to increase the number of dimensions without using communication cards with more ports?
Increasing the torus performance

- The higher number of dimensions:
  - The higher number of ports.
  - The hardware complexity increases:
    - More expensive chip production.
    - Difficulty to implement some techniques (e.g. VOQ).
  - More expensive hardware.

- Is it possible to increase the number of dimensions without using communication cards with more ports?

- **Idea**: Combine several cards as a single node communication hardware.
  - Simplest case: to interconnect two cards by one port.
Outline

Introduction

nDT torus topology

Building nDT torus using EXTOLL cards

Performance evaluation

Conclusions and future work
Example

- Given 4-port cards, a 2D torus can be built...
- Or we can use one port per card for interconnecting two cards.
- There are still 6 ports to build a 6-port node.
- A 3D torus can be built using these nodes, and it is called 3D Twin (or just 3DT) torus.
Example

- Given 4-port cards, a 2D torus can be built...
- Or we can use one port per card for interconnecting two cards.
- There are still 6 ports to build a 6-port node.
- A 3D torus can be built using these nodes, and it is called 3D Twin (or just 3DT) torus.
3DT torus topology

Example

- Given 4-port cards, a 2D torus can be built...
- Or we can use one port per card for interconnecting two cards.
- There are still 6 ports to build a 6-port node.
- A 3D torus can be built using these nodes, and it is called 3D Twin (or just 3DT) torus.
Example

- Given 4-port cards, a 2D torus can be built...
- Or we can use one port per card for interconnecting two cards.
- There are still 6 ports to build a 6-port node.
- A 3D torus can be built using these nodes, and it is called 3D Twin (or just 3DT) torus.
nDT torus topology

- This idea can be generalized for \( n \) dimensions.
- \( nD \) Twin (\( nDT \)) torus topology.
- Node communication hardware: two \((n+1)\)-port cards.
  - There are a total of \(2n + 2\) ports.
  - One port of each card interconnects the two cards.
    - We refer to this port as “internal link”.
  - There are \(2n\) remaining ports to connect the node with the neighbour nodes.
**nDT torus topology**

- This idea can be generalized for $n$ dimensions.
- $nD$ Twin ($nDT$) torus topology.
- Node communication hardware: two $(n + 1)$–port cards.
  - There are a total of $2n + 2$ ports.
  - One port of each card interconnects the two cards.
    - We refer to this port as “internal link”.
  - There are $2n$ remaining ports to connect the node with the neighbour nodes.

**Example**

- 4–port cards $\Rightarrow$ 2D torus or 3DT torus.
- 6–port cards $\Rightarrow$ 3D torus or 5DT torus.
- 8–port cards $\Rightarrow$ 4D torus or 7DT torus.
The message latency depends on the node configuration.

Crossing two internal cards, the latency increases!

\( nDT \) torus: \( \left( \frac{2n}{n} \right)^2 \) configurations.

Optimal configuration \( \Rightarrow \) Minimizes the number of paths that use the internal link.
Optimal configuration for nDT torus node

(a) $n$ is even

(b) $n$ is odd

Optimal node configuration.
Optimal configuration for nDT torus node

Example
Given 6–port cards, we can build a 5DT torus and the optimal configuration is:

▶ The ports of $d_0$ and $d_1$ are connected to the first card.
▶ The ports of $d_3$ and $d_4$ are connected to the second card.
▶ The ports of $d_2$ are separated between the two cards.
DORT routing algorithm

- DORT implements the DOR algorithm in nDT torus.
- Each PE is identified by \((n + 1)\) digits \(\langle o_0, o_1, \ldots, o_{n-1}|pe\rangle\):
  - \(o_0, o_1, \ldots, o_{n-1} \Rightarrow d_0, d_1, \ldots, d_{n-1}\) coordinates.
  - \(pe \Rightarrow\) Processing element identifier.
- First, a packet is routed from \(d_0\) to \(d_{n-1}\) dimensions.
  - If the output port does not belong to the current card, the packet is routed to the internal link.
- When the packet arrives at the destination node:
  - Destination PE = current PE? \(\Rightarrow\) route it to the NIC.
  - Destination PE = neighbour PE? \(\Rightarrow\) route it to the internal link.
- Virtual channels used in internal link to avoid deadlocks.
Outline

Introduction

nDT torus topology

Building nDT torus using EXTOLL cards

Performance evaluation

Conclusions and future work
EXTOLL cards

- EXTOLL$^1$ is an interconnection network technology designed to achieve:
  - Very low latency.
  - A high bandwidth.
  - A high sustained message rate.
  - A high availability in large-scale networks (up to 64k nodes).

$^1$http://www.extoll.de/
EXTOLL cards

▶ EXTOLL is an interconnection network technology designed to achieve:
  ▶ Very low latency.
  ▶ A high bandwidth.
  ▶ A high sustained message rate.
  ▶ A high availability in large-scale networks (up to 64k nodes).

▶ EXTOLL cards have 6 ports.

▶ EXTOLL cards support:
  ▶ 3D torus.
  ▶ Arbitrary topologies.
  ▶ 5DT torus!!!
EXTOLL architecture
EXTOLL Network

- \(IQ\) switches:
  - Multiqueue-FIFO buffers.
  - VOQ-switch to minimize the HOL-blocking.
- Virtual cut-through.
- Fine-grain credit flow-control.
- iSLIP arbiter.
- Virtual channels:
  - To avoid deadlocks.
  - To provide adaptiveness.
- Four Traffic Classes (TCs) to provide QoS.
- Table-based routing:
  - Allows to implement arbitrary topologies.
  - Each TC can have its own routing function.
TS-DOR routing algorithm

- DORT not implementable using EXTOLL cards:
  - 4 deterministic VCs required in the internal link.
  - EXTOLL only has 2 deterministic VCs.
- New deterministic routing algorithm: Twin-source Dimension Order Routing (TS-DOR)
- Combines TCs and VCs to avoid deadlocks.
- Messages routed following different dimension orders.
- Dimension order determined by the source PE.
TS-DOR routing algorithm

- The messages generated by PE0 are:
  - Injected in TC0 or TC2.
  - Routed from $d_0$ to $d_4$.
- The messages generated by PE1 are:
  - Injected in TC1 or TC3.
  - Routed from $d_4$ to $d_0$.
- Two VCs per TC required to avoid deadlock.
- Additional advantages:
  - Shorter paths than DORT.
  - Better load-balance than DORT.
Outline

Introduction

nDT torus topology

Building nDT torus using EXTOLL cards

Performance evaluation

Conclusions and future work
Simulation model

- EXTOLLsim model
  - Models the main features of EXTOLL crossbar.
  - Routing algorithms:
    - 3D torus: fully-adaptive routing + DOR.
    - 5DT torus: TS-DOR algorithm.
- Uniform traffic pattern.
- Evaluated network sizes:

<table>
<thead>
<tr>
<th>Number of PEs</th>
<th>3D torus</th>
<th>5DT torus</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>8×8×4</td>
<td>4×4×2×2×2×2</td>
</tr>
<tr>
<td>512</td>
<td>8×8×8</td>
<td>4×4×2×4×2</td>
</tr>
<tr>
<td>1024</td>
<td>16×8×8</td>
<td>4×4×2×4×4</td>
</tr>
</tbody>
</table>

- Results:
  - Normalized throughput (cells/cycle/NIC) vs Normalized injection rate (cells/cycle/NIC)
  - Network cell latency vs Normalized injection rate (cells/cycle/NIC)
Normalized throughput

Uniform traffic pattern

Normalized Avg. Throughput vs. Normalized Injection Rate

- 8 x 8 x 4
- 8 x 8 x 8
- 16 x 8 x 8
- 4 x 4 x 2 x 2 x 2
- 4 x 4 x 2 x 4 x 2
- 4 x 4 x 2 x 4 x 4
Network cell latency

Uniform traffic pattern

- 8 x 8 x 4
- 8 x 8 x 8
- 16 x 8 x 8
- 4 x 4 x 2 x 2 x 2
- 4 x 4 x 2 x 4 x 2
- 4 x 4 x 2 x 4 x 4

Normalized Injection Rate vs. Avg. Cell Network latency (cycles)
Outline

Introduction

nDT torus topology

Building nDT torus using EXTOLL cards

Performance evaluation

Conclusions and future work
Conclusions and Future Work

▶ Conclusions:
▶ To build an nDT torus is possible using the commercial hardware EXTOLL.
▶ TS-DOR: A new deadlock-free routing algorithms have been designed.
▶ We have proved the network performance is increased:
  ▶ Network latencies are reduced.
  ▶ The network accepts more traffic.
  ▶ Lower throughput degradation using virtual channels.

▶ Future work:
▶ Evaluation using other traffic patterns and MPI traffic.
▶ Performance comparison of DORT and TS-DOR algorithms.
▶ Adaptive routing for EXTOLL 5DT torus.
A case study on implementing virtual 5D torus networks using network components of lower dimensionality

HiPINEB 2017

Francisco José Andújar Muñoz et al.
fandujar@dsi.uclm.es