

1 Review “Multiprocessors and Multicomputers” (1)

Summary

This paper discusses the concepts of multiprocessors and multicomputers. It starts by outlining that multiprocessors or multicomputers which already previously could be used in high-end use cases to speed-up performance could now also be used cost effectively at scale. The paper discusses the different types of hardware architectures of multiprocessors/-computers.

- The paper differentiates between different systems. First it differentiates the application binary interface (ABI). There are shared-memory ABIs where the different threads communicate via a shared memory. There are also Messaging ABIs where the different threads communicate via a messaging system. The paper also differentiates two different system types. Multiprocessors where you have several CPUs in a single system and multicomputers where you have multiple compute nodes working in tandem.
 - Shared-memory ABI on a shared-memory multiprocessor.
 - Messaging ABI on a shared-memory multiprocessors.
 - Shared-memory ABI on a multicomputer.
 - Messaging ABI on a multicomputer.
- The paper discusses the issue of cache coherence for shared-memory ABIs. But the paper also outlines that this issue is handled by the cache coherence protocol which works regardless of the number of agents accessing the memory.
- The paper further discusses several proposed systems and their approach to solve these issues.
- The paper then concludes that multicomputers and multiprocessors have a bright future and as a consequence of them becoming cost effective will be wide spread in the future.

Strengths

The paper does not provide its own research but rather summarizes previous work. It outlines the clear strengths but also hurdles of multiprocessing systems.

Weaknesses

The paper is basically a literature review. As such it is very informative but in some cases lacks detail and depth.

My POV

It is clear that the paper was correct with its prediction pertaining to multiprocessing and multicomputing systems. As today basically all systems are multiprocessing systems and parallelization is a major source of speed-up in modern software.

Takeaways

- Multiprocessing is a powerful tool to efficiently use a compute system. But it also requires precise communication between different threads.

2 Review “CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators” (2)

Summary

The paper proposes a cache coherence protocol for near-data-accelerators (NDAs) called CoNDA. This is motivated by the increased research in NDAs and their performance benefits. The paper proposes several solutions outlining their benefits and detriments. The paper evaluates the performance of CoNDA on a diverse set of tests.

- The paper first outlines that recent trends, partially due to the advancement in 3D stacking, have shown an increase in NDA systems. Therefore the paper identifies the need for an efficient NDA cache coherence protocol.
- The paper proposes several cache coherence protocols for NDAs.
 - Non-Cachable Approach, sidestepping coherence.
 - Coarse-Grained Coherence, where a single coherence permission is used that applies to the entire NDA data region.
 - Fine-Grained Coherence, which is analogous to the traditional MESI cache coherence scheme.
- The paper concludes for these initial proposals that all have significant drawbacks and in response propose CoNDA. CoNDA is an optimistic cache coherence policy. This means that the NDA kernel executes assuming that no conflict occurs. It does not write back to the cache until it terminates execution and then checks whether an actual conflict has occurred. If that is the case this conflict is then resolved. This has the advantage of moving all coherence operations to the end of the NDA kernels execution which avoids large continuous data movement which incurs high energy cost.
- The CoNDA approach is based on the observation that most NDA kernels have very few conflicts with CPU threads and therefore an optimistic approach performs very good on average even if it has a very bad worst case performance.
- CoNDA hides the coherence from the programmer. Therefore the NDA kernel thread can be treated the same as any other CPU thread. It also provides synchronization primitives which are very similar to normal multithreaded execution.
- The paper evaluates CoNDAs hardware overhead and concludes that it only uses a minimal amount of DRAM and increases the NDA L1 cache by 1.6%.
- The evaluation of CoNDA shows a significant decrease in off-chip data movement, the best performance over all tested alternatives (not counting the ideal implementation), lowest energy consumption over all tested alternatives (not counting the ideal implementation). The paper also shows that the relative improvements hold for more stacks as well as for large datasets.

Strengths

The paper provides a simple but clear coherence protocol for NDAs that shows good performance in the used test cases. The paper also provides a thorough evaluation of the performance and data movement of the protocol.

Weaknesses

The paper only provides simulated results which therefore have to be taken with a grain of salt. As there is no previous work which can be compared to the paper only compares to a only-CPU baseline, an ideal system, and some hypothetically bad implementations. Therefore the concrete ideallity of the system cannot be evaluated fairly without having more context/research. While the average performance of CoNDA is remarkable its worst case performance is very bad. There is no evaluation of its worst case performance given in the paper.

My POV

The increase of NDA systems necessitates research of NDA cache coherence protocols. This paper is an important first step but further research is need to fairly evaluate the performance of CoNDA. The worst case performance of CoNDA could be never terminating as it might be possible that some code in the CPU could always invalidate the

NDA kernels result, mainly for a long NDA part, which could always cause a conflict at the end of execution which further always causes recomputation in the NDA. In short there is a lack of exploration of worst case performance.

Takeaways

- CoNDA is a strong contender as a NDA cache coherence protocol, but without further research this cannot be assessed fairly.

3 Review “A Case for Bufferless Routing in On-Chip Networks” (3)

Summary

The paper proposes a bufferless routing scheme for on-chip networks called BLESS. The paper discusses different versions of the BLESS scheme. The core concept is to reduce the network energy consumption with minimal impact on the systems performance, while also reducing the network buffer area.

Policies

FLIT-BLESS

- Each packet is routed independently.
- **Injection Policy:** A processor can safely inject a flit into its router when at least one incoming link is free. This allows for local flow and admission control.
- **Arbitration Policy:** The arbitration happens based on a ranking component, i.e. ranking the priority of each packet, and a port-selection component, i.e. each packet ranks the ports. The policy then assigns the highest ranking packet its preferred port, then the second highest ranking, etc.
- **Flit-Ranking:** BLESS implements a simple Oldest-First(OF) policy to rank the flits(packets). This avoids both deadlocks and livelocks.

The main issue with this system is that each flit needs to be a *head-flit* which adds overhead.

WORM-BLESS

- Packages are issued as worms where each worm has a head and a tail.
- In a router if the head of a worm is routed to a port, then this port is allocated to the entire worm until the tail is passed.
- **Injection-Policy:** The policy is the same as for FLIT-BLESS. But if an injection happens in the middle of a worm, then this worm will be truncated and the first packet in the truncated part will be labeled as head.
- **Arbitration Policy:** The arbitration policy stays the same.
- **Flit-Ranking:** The ranking is the same as for FLIT-BLESS.

BLESS with Buffers

- **Injection Policy:** The same as WORM-BLESS.
- **Flit-Ranking:** A *mustSchedule* label is introduced. Now a flit labeled *mustSchedule* will be prioritized.

For all three versions here the **Port-Prioritization** has been omitted here, it is given as table in the paper.

Advantages and Disadvantages

Advantages The paper highlights the following advantages:

- No buffers reduce both complexity and energy consumption.
- Local and simple flow control
- Simplicity and router latency reduction
- Area saving
- Absence of deadlocks
- Absence of livelocks

Disadvantages The paper highlights the following disadvantages.

- Increased latency and reduced bandwidth
- Increased buffering at the receiver
- Header transmission with each flit for FLIT-BLESS

The paper backs this up with data collected on a simulated model of the BLESS routing scheme.

Strengths

The paper has strong fundamentals and the core idea of decreasing complexity and energy consumption by using a buffer-less routing scheme is a very smart approach. The paper clearly outlines both the advantages and disadvantages of BLESS.

Weaknesses

The paper only provides simulated results which have to be taken with a grain of salt. Otherwise the paper is very well founded and discusses both the advantages and disadvantages in sufficient detail.

My POV

The decision of whether to use a buffer-less routing scheme like BLESS is clearly a tradeoff. The paper clearly provides the tradeoffs which can be made. In other words the paper is well written and critical.

Takeaways

- Buffer-less routing schemes like BLESS have advantages and disadvantages. Therefore it is important to consider the possible tradeoffs when selecting an option in a real system and take into account the priorities and bottlenecks of the whole system.

4 Review “Virtuoso: Enabling Fast and Accurate Virtual Memory Research via an Imitation-based Operating System Simulation Methodology” (4)

Summary

The paper introduces a simulation framework to simulate virtual memory called Virtuoso. Virtuoso is designed to occupy the design space between full, low level, OS simulators and high-level simulation tools. The goal is to allow for accuracy similar to low level simulators while keeping the speed of high level simulators. The paper demonstrates the fundamental design of Virtuoso and provides its verification and a performance analysis.

Virtuoso

- Virtuoso uses an operating system MimicOS which is a lightweight kernel very similar to the Linux Kernel.
- Virtuoso works by offloading instructions in the simulation space to a MimicOS binary to execute which returns the relevant metadata and allows Virtuoso to produce the relevant simulation results.
- By offloading to MimicOS Virtuoso can be configured to only support the OS functionalities which are necessary to handle the workload. This enables a tradeoff between enabled kernel modules and simulation cost.
- Virtuoso works with a userspace kernel which can be written in a high-level language like Python or C++. This allows for easy development and fast testing cycles.
- Virtuoso establishes two communication channels between the kernel and the simulator. The first is a functional and the second an instruction stream channel.
- Virtuoso uses synchronization primitives to achieve high simulation speed while maximizing portability.
- Virtuoso’s userspace kernel supports multithreading.

MimicOS

MimicOS is a lightweight userspace kernel for memory management.

- MimicOS uses a memory management scheme similar to that of the Linux kernel.
- The paper integrated MimicOS with four architectural simulators: Sniper, Ramulator2, ChampSim and gem5-SE. It also integrated it with an SSd simulator MQSim.

Limitations

Virtuoso is a good fit for studies focusing on VM, which spans across hardware and OS layers of the system stack. Virtuoso provides greater speed than architectural simulators while maintaining high speed. But it is not designed to replace them. Virtuoso enables rapid testing of concepts and ideas but in many cases a low level implementation is still needed to find possible performance bottlenecks or to evaluate the performance of a system.

Strengths

Virtuoso is a tool with a clear use case. Its relevance stems from its acceleration of the testing loop. Virtuoso allows for high-level language integration into a relatively low-level simulation. This allows for fast and easy results on complex concepts without needing expertise in kernel design.

Weaknesses

I do not see any clear weaknesses in this paper. It clearly states its goals as well as its limitations. It also provides results to validate its claims.

My POV

Virtuoso is a valuable tool that allows rapid development cycles and high versatility.

Takeaways

- When testing a concept or novel idea the choice of simulation tool has a great impact on development speed. Virtuoso allows for fast development cycles for high-level and early stage concepts.
- Virtuoso is not intended to replace low-level simulators and an architectural simulator is still necessary to provide final performance results.

5 Review “Validity of the single processor approach to achieving large scale computing capabilities” (5)

Summary

This paper is a discussion about the validity of the single processor approach versus the multiprocessor approach. The paper outlines that a certain part of the runtime of any program has to be computed serially. The paper further states that to be able to harness the positive effects of parallel computing one must equivalently improve the serial components of the computation. The paper introduces a law which relates the speedup to the inverse of the serial part times the parallel part divided by the number of parallel threads.

Strengths

The paper highlights the issues with parallel computing and the prevalent issues at the time of writing the paper. It also introduces a law to estimate the possible speed-up of a parallel system.

Weaknesses

While the paper was clear in its deductions and was valid at the time, it is clear now that parallel computing is the future. While this is still in line with the predictions of the paper, i.e. the serial part of computation has been improved significantly, it is still important to state that the overall conclusion of the paper has been proven false.

My POV

The paper is dated but introduced the important concept of Amdahl's Law which governs parallel computation to this day. While the progress has outgrown the predictions made in this paper it is still an important piece of research.

Takeaways

- Any parallel process always has a serial component which cannot be disregarded. The system performance can be computed with Amdahl's Law

6 Review “Route Packets, Not Wires: On-Chip Interconnection Networks” (26)

Summary

The paper gives an overview of on-chip interconnection networks. The paper starts of by outlining the concept of interconnection networks.

- The chip is split into routers, dependent on the on chip system this could for example be individual CPUs in a multiprocessor network.
- Each router provides in- and output-ports to handle incoming and outgoing data.
- Data is sent in packets, which start with a head and end with a tail. Metadata is passed with the packet to allow each router to dynamically decide on the routing and prioritization of the packets.

The paper suggests a system with a simple interface that allows for customization of higher level protocols. The paper further discusses different advantages protocols could provide on a network.

- Fault-tolerant protocols which allow identification of faulty bits in the transmitted data.
- Pre-scheduling protocols which allow predictable data to be scheduled ahead of time and then to be combined with dynamic traffic in the moment.

The paper also discusses different network topologies such as a simple grid or mesh topology, which provides an even distribution but high pin count. Also the torus topology is proposed to allow for fewer routing hops at the cost of longer transmission time.

Lastly the paper argues that on-chip networks while they do incur a significant area overhead, estimated at 6.6%, their benefits in terms of reduction of parasitics and increase in routing efficiency outweigh these detriments.

Strengths

The paper provides a clear and strong vision for the implementation of on-chip interconnection networks in processors. The proposed routing system provides a resilient and dynamic approach to the on-chip communication.

Weaknesses

There are no big weaknesses in this paper except that clearly further research was still required at time of publishing to allow this technology to mature.

My POV

On-chip interconnection networks are common on modern multiprocessors and similar systems. In other words, the paper correctly estimated the incredible effectiveness of on-chip networks. The paper is not very specific in its analysis and gives a high level overview of the concept. Therefore there is not much to criticise or improve on.

Takeaways

- On-chip interconnection networks are here to stay and provide an efficient solution to the problem of on-chip communication.

7 Review “A QoS-Enabled On-Die Interconnect Fabric for Kilo-Node Chips” (30)

Summary

The paper outlines the issue for larger chips with hundreds to thousands of on-chip components that require communication old network-on-chip designs don't work well due to high area, energy, and buffer overheads. This issue is especially pronounced when quality-of-service (QoS) guarantees are needed. The paper thus outlines Kilo-NoC which addresses the issues as follows:

- The paper introduces a hybrid NoC architecture that co-optimizes topology, flow control, and QoS by creating subnetworks and providing interconnections between these subnetworks.
- The paper introduces a topology aware QoS scheme that limits hardware QoS support to the dimensionality requirements of the packets.
- The paper introduces a lightweight elastic buffer (EB) flow control that embeds buffering in links which minimizes the overall buffer cost.

The paper also evaluates Kilo-NoC its results show a substantially lower area and energy compared to state-of-the-art QoS-enabled NoCs, while maintaining strong service guarantees.

Strengths

The proposed splitting of the interconnection networks into subnetworks or different dimensions is a strong contender to allow efficient routing as a lot of routing happens locally. The elastic buffer approach greatly diminishes buffer storage requirements, which are a major cost driver in richly connected topologies.

Weaknesses

The proposed solution comes with a high complexity which can draw significant cost in an actual physical implementation. The paper also only provides simulated results which always have to be taken with a grain of salt.

My POV

The paper makes a strong case for complex interconnection topologies to handle large on-chip interconnection networks. Similar technology is in use in modern multiprocessors which clearly emphasises strength of some of the ideas outlined in the paper. It would still be interesting to see the actual results of the network if they had been physically implemented.

Takeaways

- Complex, high component chips require novel solutions to routing. This is an issue that will not lessen in the future but become more prevalent as even today the number of components on a chip increases.