Co-Optimization of Memory Access and Task Scheduling on MPSoC Architectures with Multi-Level Memory

Dr. Chun Jason Xue
Assistant Professor
Computer Science
City University of Hong Kong
http://www.cs.cityu.edu.hk/~jasonxue/
jasonxue@cityu.edu.hk

Outline

Introduction

Example

Integer Linear Programming

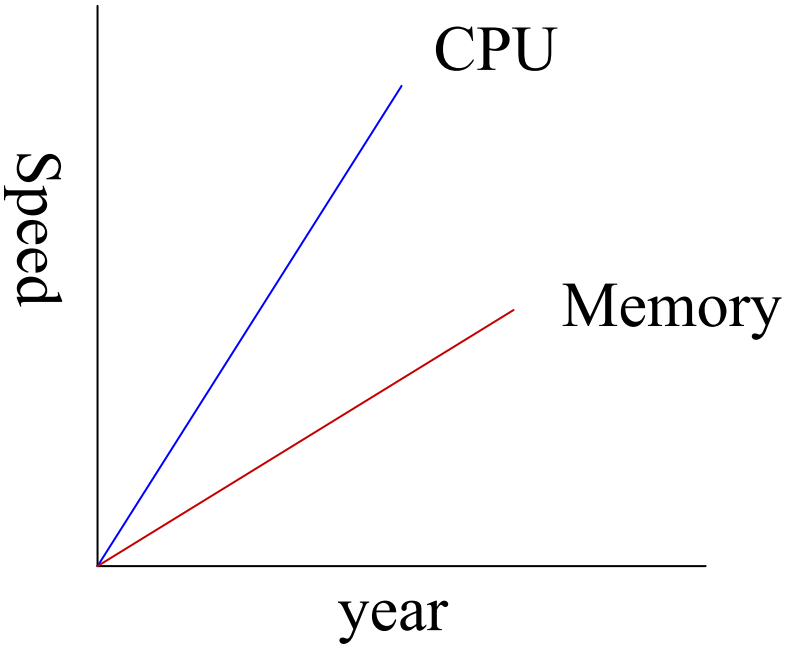
LPC Algorithm

Experiments

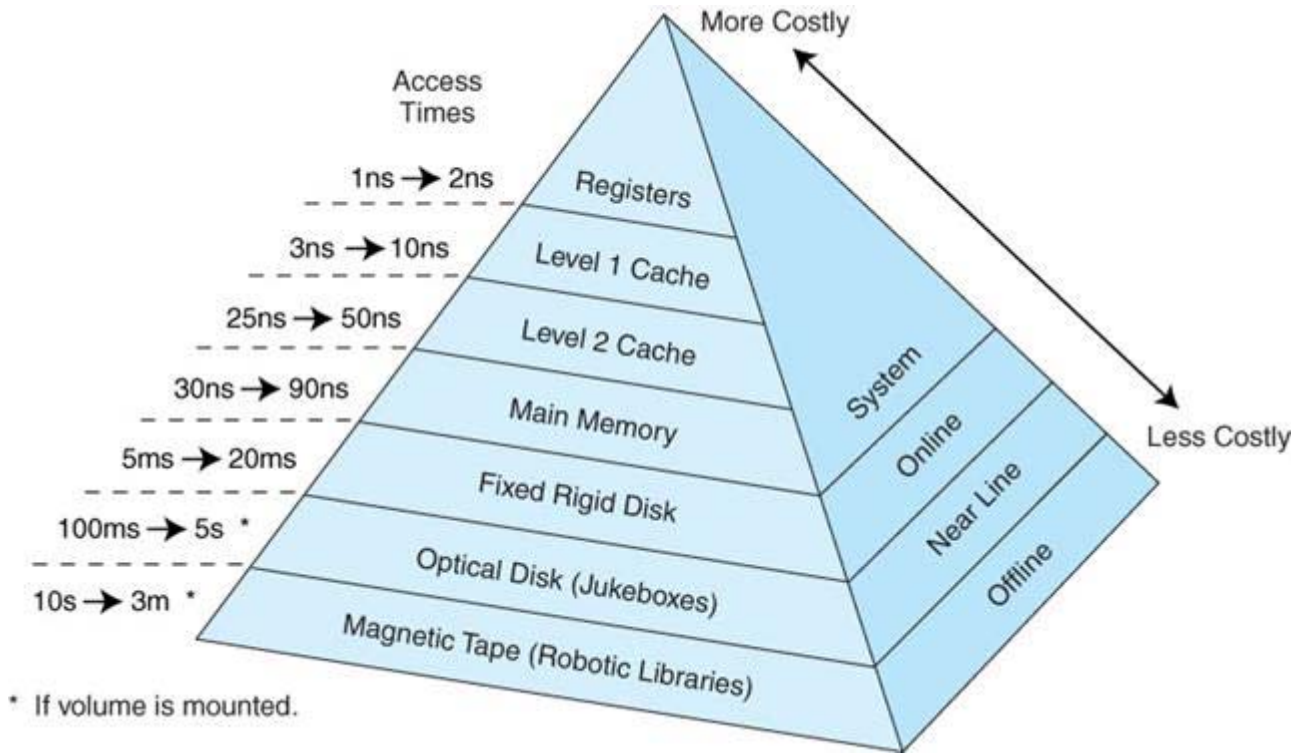
Conclusion

Memory Wall

The rate of improvement in microprocessor speed exceeds that of memory: Processor – Memory Performance GAP



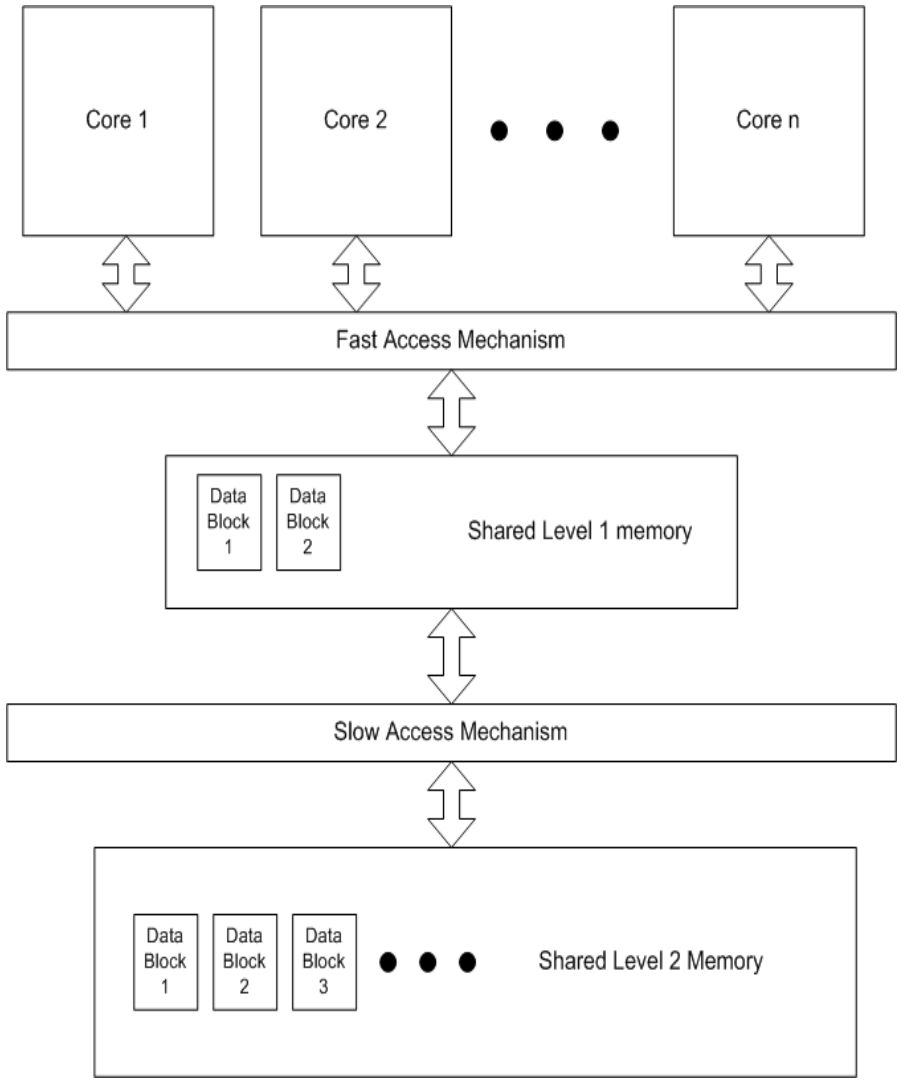
Memory Hierarchy



Dilemma

- The higher in the pyramid, the faster the access speed, but… the higher, the more expensive.
- A lot of embedded systems have multi-level memory systems (just like our home computers)
- A trade-off must be made between speed and cost

A Multi-processor System



- N cores: N computation units
- There are 2 levels of memory
- Level 1 is much fast than level 2, but it does not have all the space for data

MPSoC applications

- Possible applications: real-time DSP, video processing, multimedia implementations, etc…
- A majority of these applications require intensive memory access during computation.
- A fundamental problem: how to get the best schedule that can take the advantage of locality?
- We attack this problem from the aspect of scheduling.

Contributions

- Both scheduling and memory accesses are considered during the optimization of the schedule.

- The proposed algorithms can be applied to all systems with multi-level memory, especially those targeting memory access intensive applications.

- An ILP model is proposed to achieve the optimum schedule, and a heuristic algorithm is developed if the running time is required to be polynomial.

Outline

Introduction

Example

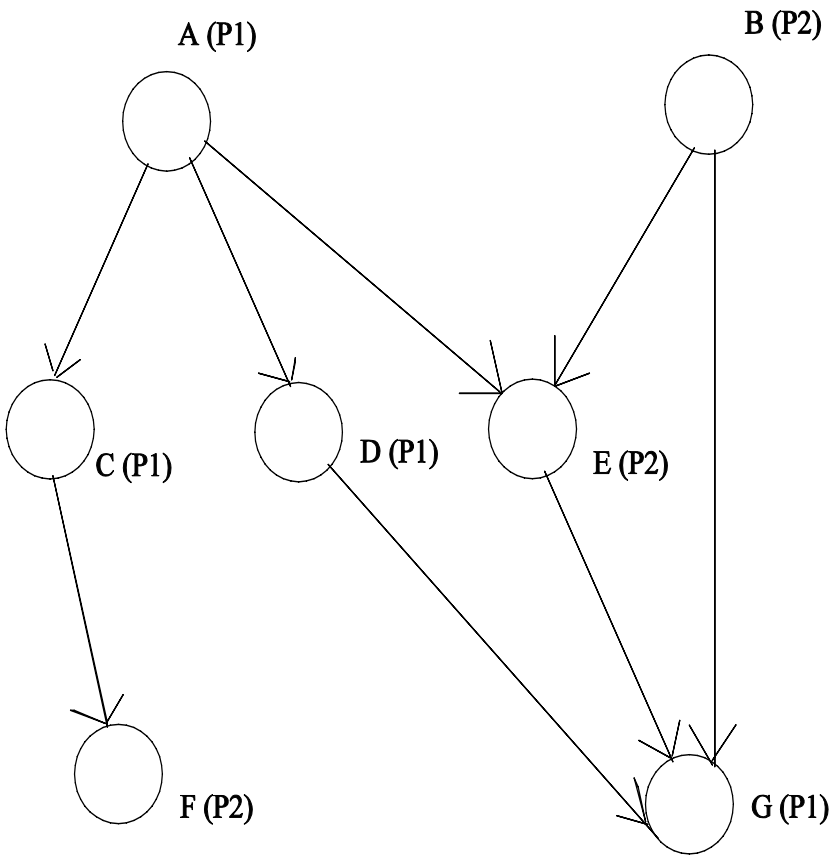
Integer Linear Programming

LPC Algorithm

Experiments

Conclusion

Motivational Example



- Each node is a block of computations that can be grouped together. (A, B, C,…)
- The arrows stand for dependencies between nodes.
- P1, P2 are the two data blocks needed for the computations

Schedule 1

Step	Node	Page Accessed	Load in L1	Data access Time (units)
1	A	P1	Yes	50
2	B	P2	Yes	50
3	C	P1	Yes	50
4	D	P1	No	5
5	E	P2	Yes	50
6	F	P2	No	5
7	G	P1	Yes	50

Schedule 2

Step	Node	Page Accessed	Load in L1	Data Access Time
1	A	P1	Yes	50
2	C	P1	No	5
3	D	P1	No	5
4	B	P2	Yes	50
5	E	P2	No	5
6	F	P2	No	5
7	G	P1	Yes	50

Comparison of the 2 schedules

- In Schedule 1, there are 5 memory load operations from Level 2 memory to Level 1 memory and the total data access time is **260**.

- In Schedule 2, there are just 3 page load operations and the corresponding data access time is **170**.

- The second schedule has **35%** improvement over the first one in data access time.

Outline

Introduction

Example

Integer Linear Programming

LPC Algorithm

Experiments

Conclusion

Symbols

- X_{ij}: binary variable, whether task i is scheduled at step j.
- P_{kj}: binary variable, whether page k is in the level 1 memory at step j.
- Z_{kj}: binary variable, whether page k is loaded into the level 1 memory at step j.
- u_{ij}: a real variable that's between 0 and 1.

Integer Linear Programming

- Objective function: minimize page load

$$\text{minimize } \Sigma \text{ cost} * X_{ij.}$$

- Constraints:

1. Operation mapping: A task is scheduled only once on one core at one step.

$$\Sigma X_{ij} = 1, \ \forall i.$$

2. Resource constraints:

- At any step, the number of tasks scheduled should be at most the number of cores :

$$\Sigma X_{ij} \leq \text{cores}, \ \forall j.$$

- At any step, the number of pages in the level 1 memory should not exceed its capacity:

$$\sum P_{ij} \leq \text{capacity}, \quad \forall j.$$

3. Dependency constraints: the dependencies among tasks must be observed.

$$\sum X_{il} * l \leq \sum X_{jl} * l - 1, \quad \forall (i, j) \in E.$$

4. Causal constraints: If a task is scheduled at a specific step, the pages that it accesses must be in the level 1memory at that step.

$$c+1 \geq u_{ij,}$$

$$-X_{ij} + 1 \geq u_{ij,}$$

$$P_{kj} \geq 1 - u_{ij,}$$

$$P_{ij} - P_{i(j-1)} \geq 3 * Z_{ij}.$$

ILP analysis

- Schedule construction: once the ILP problem is solved, the corresponding schedule can be constructed by following the results of X_{ij}, P_{kj}, Z_{kj}. Node i is scheduled at step j if X_{ij} =1. If P_{kj}=1, page k is in the level 1 memory at step j. If Z_{kj}=1, there is a memory load for page k at step j.

- The achieved schedule is optimum.

- The running time could be exponential.

Outline

Introduction

Example

Integer Linear Programming

LPC Algorithm

Experiments

Conclusion

The Least Projected Cost Algorithm

- Step 1, put all nodes without dependencies into the ready list.
- Step 2, for each node in the ready list, calculate the projected cost (the minimum number of steps that another unscheduled task accesses the same page).
- Step3, find the node with the minimum projected cost and schedule it.
- Step 4, try all positions in the existing schedule and find out the best legal position so that the page load in the schedule is as minimal as possible.

LPC algorithm (continued)

- Step 5, mark the node as scheduled, update the schedule and the ready list.
- Repeat from step 2.
- Step 6, repeat this process until all nodes are scheduled.

LPC algorithm analysis

- This is a heuristic algorithm.
- The running time is polynomial.
- It's a greedy algorithm. At each step, how to choose the next node to be scheduled is based on an estimate of how far ahead that the same page is going to be accessed again. During scheduling , the node is inserted in the best position possible at that time.
- For applications with a large number of nodes, this algorithm is better suited.

Outline

Introduction

Example

Integer Linear Programming

LPC Algorithm

Experiments

Conclusion

Experiments: DSP benchmarks

Benchmark (# nodes)	List	ILP		LPC	
2iir (16)	27	18	33%	18	33%
deq (11)	15	9	40%	12	20%
4latiir (26)	36	21	42%	24	33%
volt (27)	39	21	46%	30	23%
elf (34)	48	21	56%	27	44%
8latiir (42)	63	*	*	45	29%
Avg. %	n/a	n/a	43%	n/a	30%

Results analysis

- Both algorithms surpass list scheduling. ILP can achieve the best results among these 3. LPC can achieve close to optimum results in most cases.

- ILP has better results in memory load cost as long as it can be computed. However, in some cases, it simply takes too long to compute the ILP solution (e.g. 8latiir). Thus LPC algorithm shall be used in these kind of cases.

Outline

Introduction

Example

Integer Linear Programming

LPC Algorithm

Experiments

Conclusion

Conclusion

- Scheduling matters when it comes to memory access time
- When the schedules are optimized based on memory access, the memory access time can be dramatically lowered
- Memory access intensive applications will benefit

Q & A ?

Thank You!