Parallel Execution with Data-Driven Multithreading

Data-Driven Multithreading (DDM)

DDM can:
- Hide communication latencies
- Hide synchronization latencies
- Exploit large amount of parallelism

Implementation on a CMP (DDM-CMP)
- Commodity CPUs
- Commodity OS
- Commodity Compiler
- Programming the DDM C preprocessor
- Full system simulation with Simics

The DDM-CMP multiprocessor

The DDM-CMP multiprocessor consists of multiple CPUs with associated caches and shared memory. The system network is used to connect the processors and main memory. Each thread has access to its own local cache and can access shared memory through the system network. The Shared Main Memory provides a unified memory space for all threads.

Experimental Results

The experimental results demonstrate the performance improvements achieved with DDM-CMP compared to a baseline system. The graphs show the speedup for different problem sizes and configuration settings, highlighting the benefits of data-driven multithreading in terms of both execution speed and resource utilization.

Future Directions

Operating System optimizations
- Currently: Unmodified OS
- To Study:
  - DDM applications/OS Interaction
  - Avoid Context switches

Data Prefetching
- The next thread is known
- Its data can be **prefetched**
- Cache modifications
- TSU triggered prefetching
- Efficient Data prefetching

Multi-chip DDM
- Hierarchical DDM-CMP configuration
- Shared Memory inside the chip
- Data shipping across chips
- **Scalability**

This project is funded by the Cyprus Research Promotion Foundation.

Kyriakos Stavrou
pedro@cs.ucy.ac.cy
Paraskevas Evripidou
skevos@cs.ucy.ac.cy
Parallel Execution with Data-Driven Multithreading

Kyriakos Stavrou    Pedro Trancoso
Paraskevas Evripidou
University Of Cyprus
Computer Science Department
www.cs.ucy.ac.cy/carch/casper

Data-Driven Multithreading (DDM)

DDM can:
- Hide communication latencies
- Hide synchronization latencies
- Exploit large amount of parallelism

Implementation on a CMP (DDM-CMP)
- Commodity CPUs
- Commodity OS
- Commodity Compiler
- Programming the DDM C preprocessor
- Full system simulation with Simics

The DDM-CMP multiprocessor

The application's code

```
initializations
#pragma ddm parfor reduction (+:area)
for(i=0;i<ITER;i++)
    area+=f(i,i+step)
#pragma ddm end parfor
print (area)
```

The synchronization graph is automatically created

1. initializations
2. for(i=0;i<ITER/2;i++)
   area_1+=f(i,i+step)
3. area+=area_1
4. for(i=ITER/2;i<ITER;i++)
   area_2+=f(i,i+step)
5. area+=area_2

```
Thread | Iterations
--- | ---
3 | 0 1 2 3 ... 31
```

Thread completed
Inform its consumers

Experimental Results

- Trapezoidal rule for integration (sin(x))
- Matrix multiplication (double)

Future Directions

**Operating System optimizations**
Currently: Unmodified OS
To Study:
- DDM applications/OS Interaction
- Avoid Context switches (apps. merging)

**Data Prefetching**
- The next thread is known
- Its data can be **prefetched**
- Small cache modifications
- TSU triggered prefetching

**Multi-chip DDM**
- Hierarchical DDM-CMP configuration
- Shared Memory inside the chip
- Message passing across chips

Efficient Data prefetching
Scalability

This project is funded by the Cyprus Research Promotion Foundation