Have Your Cake In Parallel And Eat It Sequentially Too!

Semantically Sequential, Parallel Execution of Multiprocessor Programs

Gagan Gupta

Summary

Multiprocessors are ubiquitous, but programming them continues to be challenging

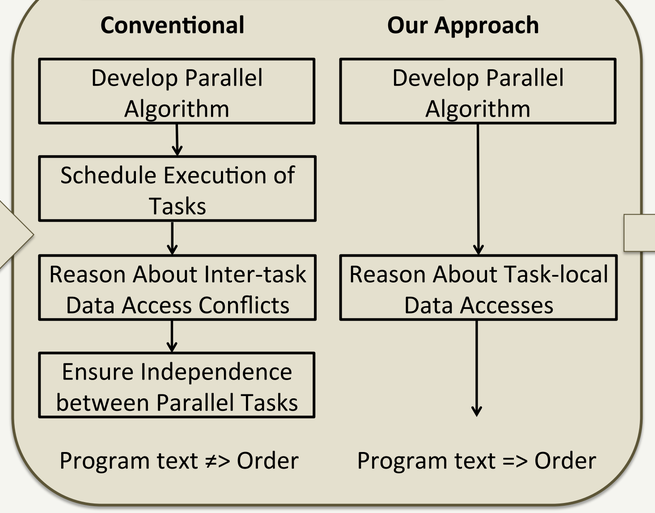
Our Goal: Simplify multiprocessor programming without compromising performance

Conventional Wisdom	Our Approach
Order in programs obstructs parallelism	Order can help to expose parallelism!
Use non-deterministic programs, or make dataflow in programs explicit	Use **ordered** programs; maintain precise **program-order** execution **semantics**
Programmer should expose parallelism	Use **run-time dataflow** and **speculative** techniques to expose parallelism

Benefits:

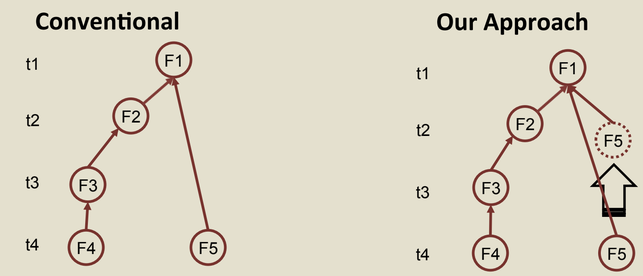
- Simplified programming; Simplified system design; Better reliability
- Performance at par or better (5% to 288%) than conventional methods

Programmer's Role



Exploiting Parallelism

Task Dependence Graph of Cholesky Decomposition

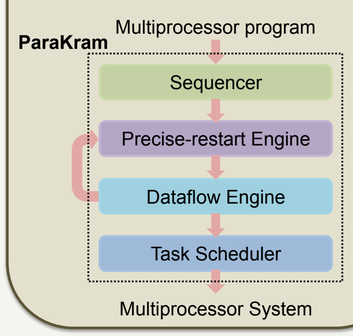


- Execution has to respect programmer-exposed parallelism
- Cannot schedule F5 in t2

- If dependences are known, distant parallelism can be exposed
- Can schedule F5 in t2

ParaKram

- Run-time parallel execution manager (C++ library)
- Performs out-of-order superscalar processor-like execution on multiprocessors



Multiprocessor Program

Sequencer

- Unrolls dynamic instances of tasks
- Computes data set dynamically (user assisted)

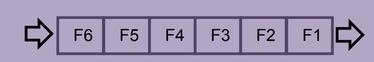
```
1  for (i=0; i<n; i++) {
2    call F (wr, rd)
3  }
```

Fn	Write Set	Read Set
F1:	{B, C}	{A}
F2:	{D}	{A}
F3:	{?}	{?}
F4:	{B}	{D}
F5:	{B}	{D}
F6:	{G}	{H}

Example code and dynamic task instances

Precise-restart Engine

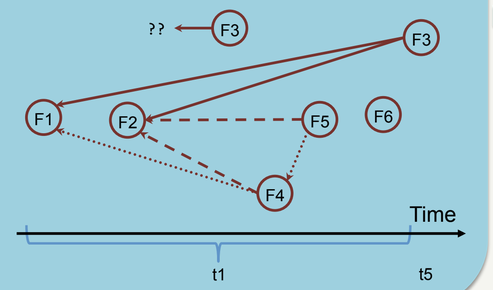
- Tracks tasks and their order in a Reorder List
- Checkpoints mod set in History Buffer
- Retires task in (total) program order



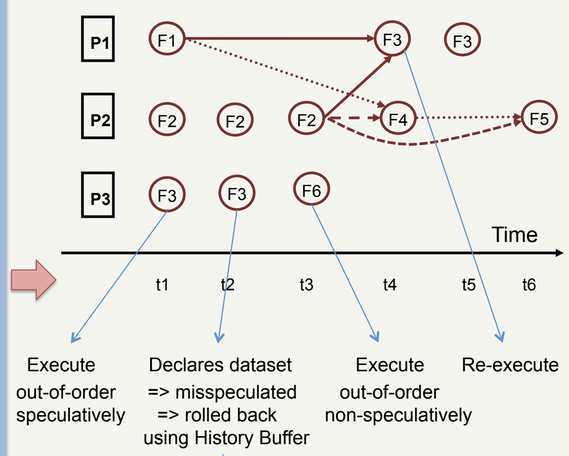
Dataflow Engine

Uncovers parallelism past blocked tasks in the program

- Constructs dynamic data dependence graph using write and read sets
- Executes tasks out-of-order
- If task dependences/order are unknown, speculates tasks are independent
- Detects and rectifies misspeculation



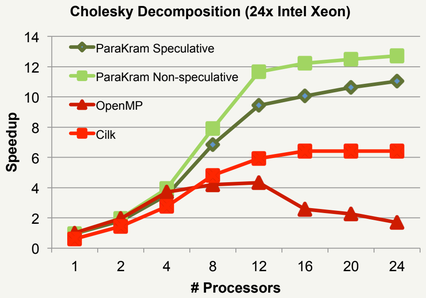
Order-aware Load-balancing Task Scheduler



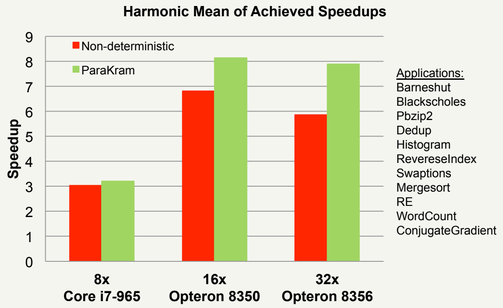
Example speculative dataflow execution on 3 processors

Epoch	Reorder List Entries	Completed	Retired
t1	F6 F5 F4 F3 F2 F1		
t2	F6 F5 F4 F3 F2	F1	F1
t3	F6 F5 F4 F3 F2	F1	
t4	F6 F5 F4 F3	F1, F6, F2	F2

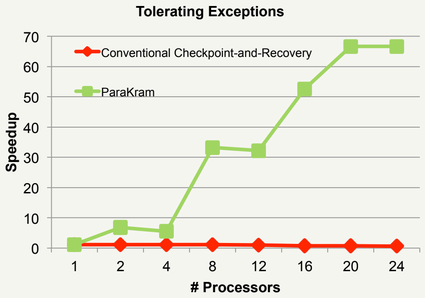
Precisely restarting misspeculated task (F3 from above)



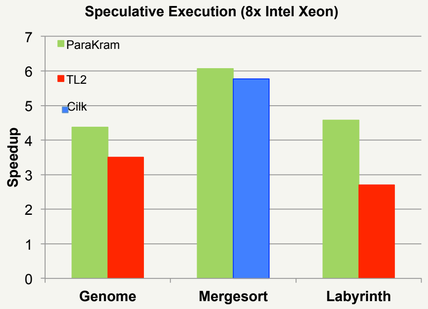
ParaKram speedup is 288% higher than non-deterministic OpenMP, 75% over Cilk



ParaKram speedup (harmonic mean) is 20% higher than non-deterministic Pthreads (excludes Cholesky)



ParaKram scales with system size; Non-deterministic method does not scale



ParaKram speedup is up to 77% higher than non-deterministic Cilk and TL2 STM