Efficient Multiprogramming for Multicores with SCAF

Timothy Creech, Aparna Kotha, Rajeev Barua

Department of Electrical and Computer Engineering, University of Maryland, College Park, MD

Introduction

- Multiprogramming serial programs on interactive systems is easy
  - time-sharing works well for serial processes
- However, hardware and programs are becoming increasingly parallel
  - fine-grained time-sharing *doesn't* work well for multiprogramming parallel processes
- **SCAF** is a runtime system for multiprogramming as *space-sharing*
  - space-sharing frees the OS from needing to time-share hardware contexts
  - SCAF reasons continuously at runtime about processes' parallel efficiencies in order improve system efficiency

Why is Multiprogramming Parallel Processes Hard?

- parallel runtimes assume the entire machine is available for each process, resulting in many more threads than hardware contexts
- operating systems simply attempt to schedule all of the threads to hardware contexts in an uncoordinated fashion
- multiprogramming parallel programs is needlessly complicated for the users

Motivating Example

- NPB OpenMP Benchmark "`lu.B.x`"
  - SunOS 5.11, Sparc T2
  - **64** threads per CPU
- Say multiple users are running the "`lu.B.x`" program on this SMP system
- Each instance of "`lu.B.x`" runs on **64** threads
- How does time-sharing handle this?

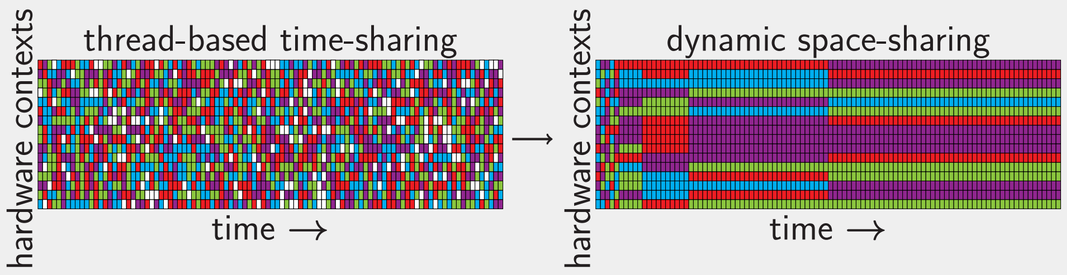
"`lu.B.x`" Instances	Machine subscription	Threads/process	Run time (s)	$\sum$ Speedup
1	100%	64	180	24.2
2	200%	64	3522	2.5
3	300%	64	8770	1.5

  - Massive oversubscription can lead to terrible system efficiency
- instead of time-sharing 64 × N threads, we would like to *space-share*

"`lu.B.x`" Instances	Machine subscription	Threads/process	Run time (s)	$\sum$ Speedup
1	100%	64	180	24.2
2	100%	32	312	28.6
3	100%	21	382	35.1

Time-sharing vs Space-sharing

- To avoid oversubscription and prevent the OS thread scheduler from time-sharing hardware contexts, we can implement space-sharing
- How many hardware contexts does each process get?
  - **equipartitioning** gives each process an equal number of contexts, but this is wasteful if a process scales poorly
  - **SCAF** reasons about per-process parallel efficiency in order to optimize the speedup gained by each hardware context



SCAF Runtime Features

- Controls multiprogrammed parallel processes' allocations to effect space-sharing
- Allocations vary at runtime in order to improve system efficiency
- Requires no offline profiling or measurements
- Requires no program porting, modification, or recompilation
  - Processes supported via modified runtime libraries (e.g. `libgomp`, `libiomp5`, etc.)
- Ports available:
  - x86 Linux: GNU OpenMP
  - Sparc SunOS: GNU OpenMP
- Ports in progress:
  - x86 Linux: Intel OpenMP
  - Tile-Gx Linux: GNU OpenMP
  - Xeon Phi Linux: GNU OpenMP, Intel OpenMP

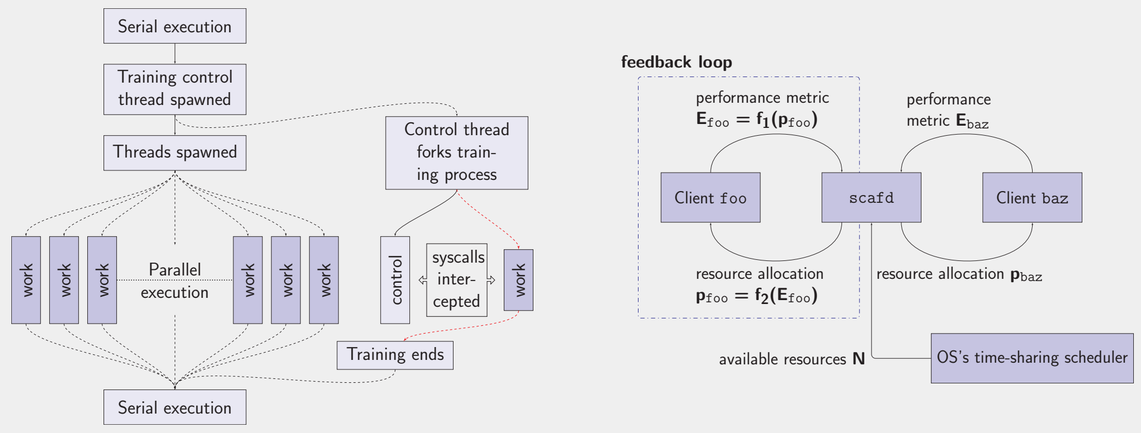
Techniques Used

Serial "experiments" at runtime

- estimate serial performance without serialization/profiling
- serial performance then compared to observed performance to reason about acheived efficiency

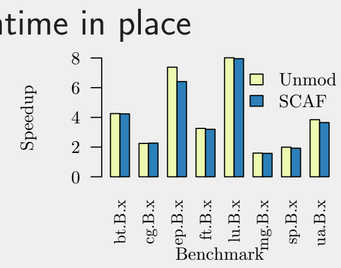
Dynamic Allocation

- malleability allows allocations to vary at runtime based on observed efficiency
- more efficient applications rewarded with more threads



Results: Minimal Overhead

- Extra logic required to implement SCAF does not adversely affect performance when processes run alone
- Results below compare executables running with and without the SCAF runtime in place

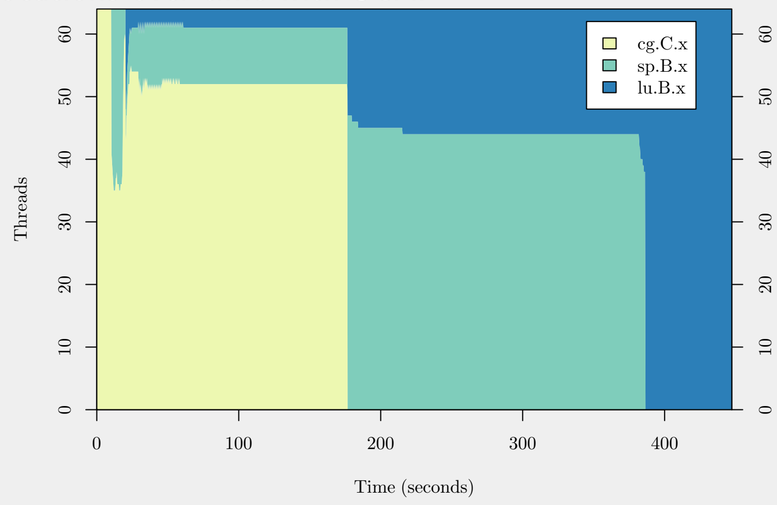


- Linux on Dual Xeon E5410

- SunOS 5.10 on UltraSparc T2

Results: Selected 3-Way Multiprogramming Example

- Multiprogramming 3 NAS benchmarks: CG, SP, and LU
- Process allocations over time with SCAF

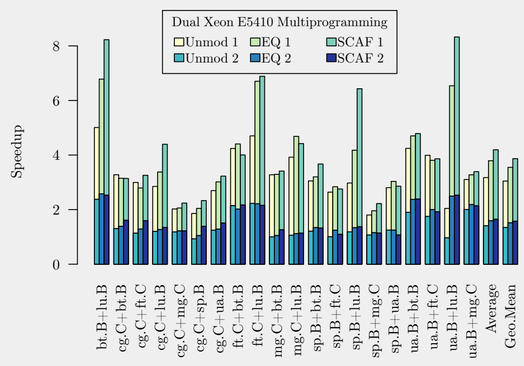


- Summary of results:

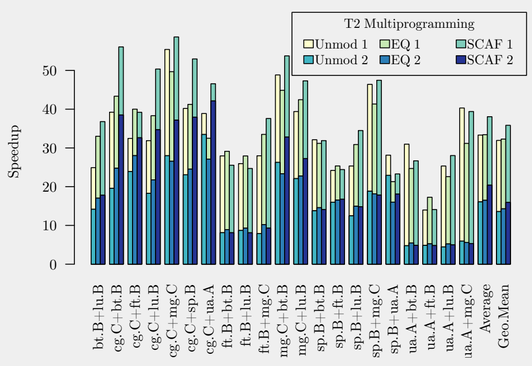
Configuration	Process	Runtime	Speedup	$\sum$ Speedup
Unmodified	CG	435.9s	13.1	31.5
	SP	474.6s	9.6	
	LU	507.3s	8.8	
Equi-partitioning	CG	374.0s	15.5	40.7
	SP	380.8s	12.2	
	LU	349.8s	13.0	
SCAF	CG	172.2s	35.7	59.3
	SP	374.0s	12.5	
	LU	424.0s	11.1	

Results: NAS Parallel Benchmarks

- 2-way multiprogramming across all NAS benchmarks on 2 platforms



- 70% of pairs see improvements over equipartitioning
- 15% average improvement



- 57% of pairs see improvements over equipartitioning
- 15% average improvement

This work was supported by a NASA Office of the Chief Technologist's Space Technology Research Fellowship.

Contact: `tcreech@umd.edu` • `http://ece.umd.edu/~tcreech`