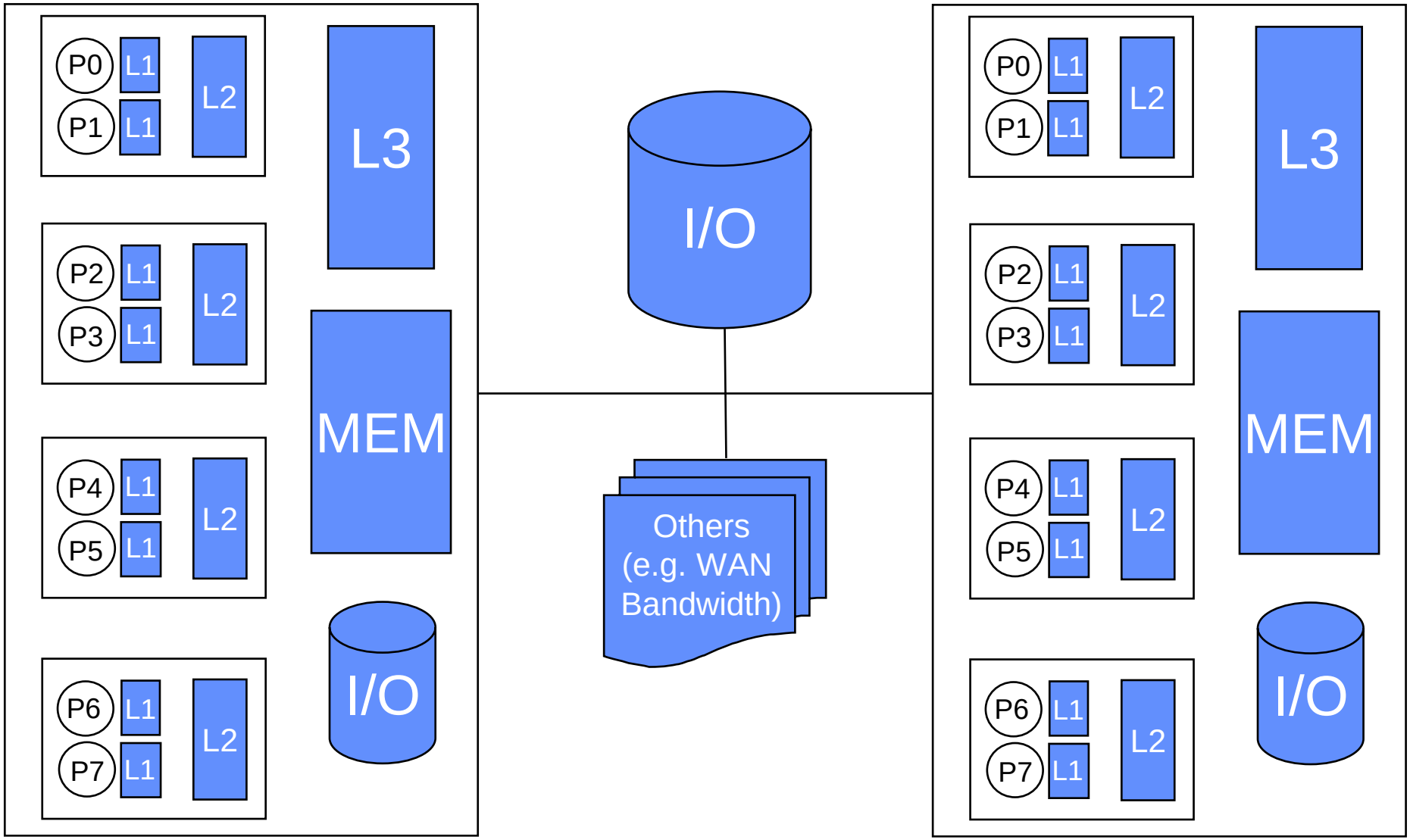
Symbiotic Space-Sharing: Mitigating Resource Contention on SMP Systems

Jonathan Weinberg
Allan Snavely

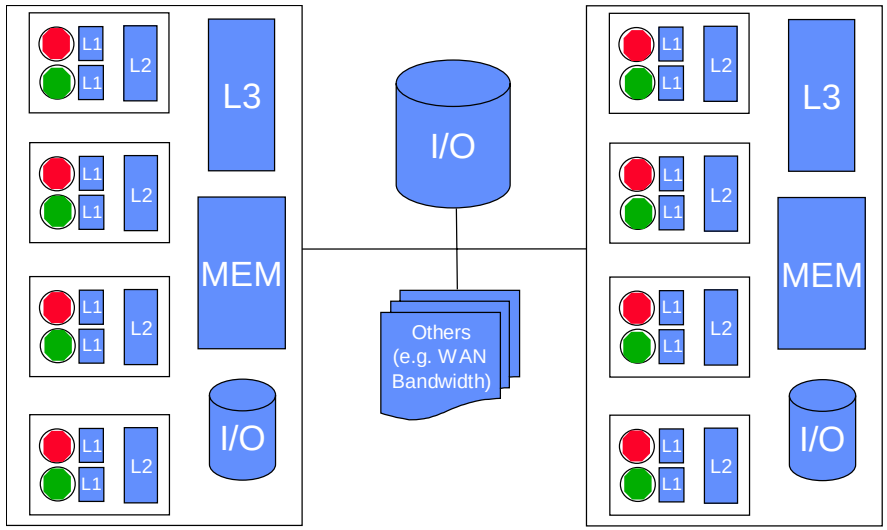
University of California, San Diego
San Diego Supercomputer Center

Resource Sharing on DataStar



Symbiotic Space-Sharing

- ***Symbiosis:*** from Biology meaning the graceful coexistence of organisms in close proximity
- ***Space-Sharing*****:** Multiple jobs use a machine at the same time, but do not share processors (vs time-sharing)
- ***Symbiotic space-sharing***: improve system throughput by executing applications in *symbiotic* combinations and configurations that alleviate pressure on shared resources



Can Symbiotic Space-Sharing Work?

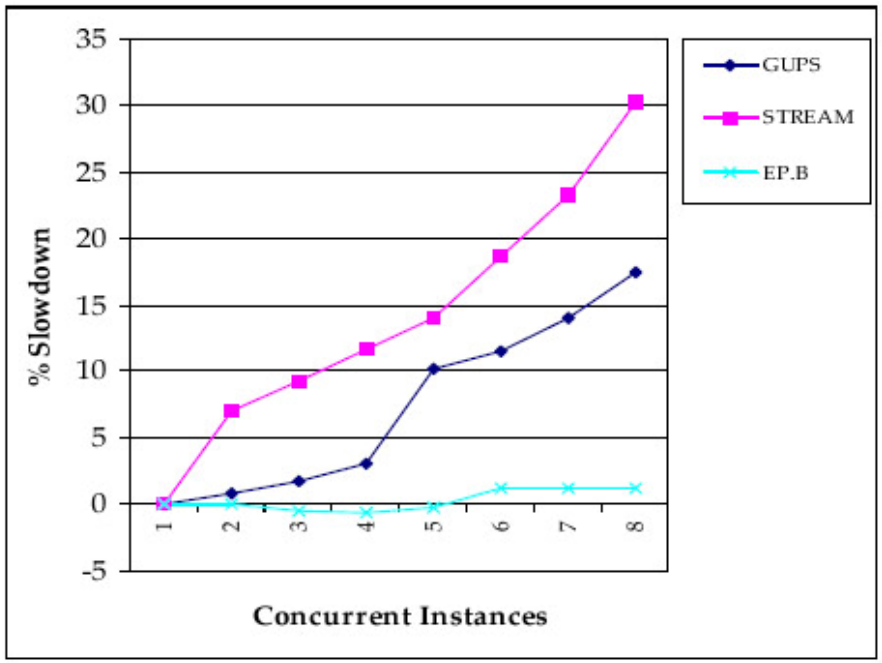
- To what extent and why do jobs interfere with themselves and each other?
- If this interference exists, how effectively can it be reduced by alternative job mixes?
- How can parallel codes leverage this and what is the net gain?
- How can a job scheduler create symbiotic schedules?

Resource Sharing: Effects

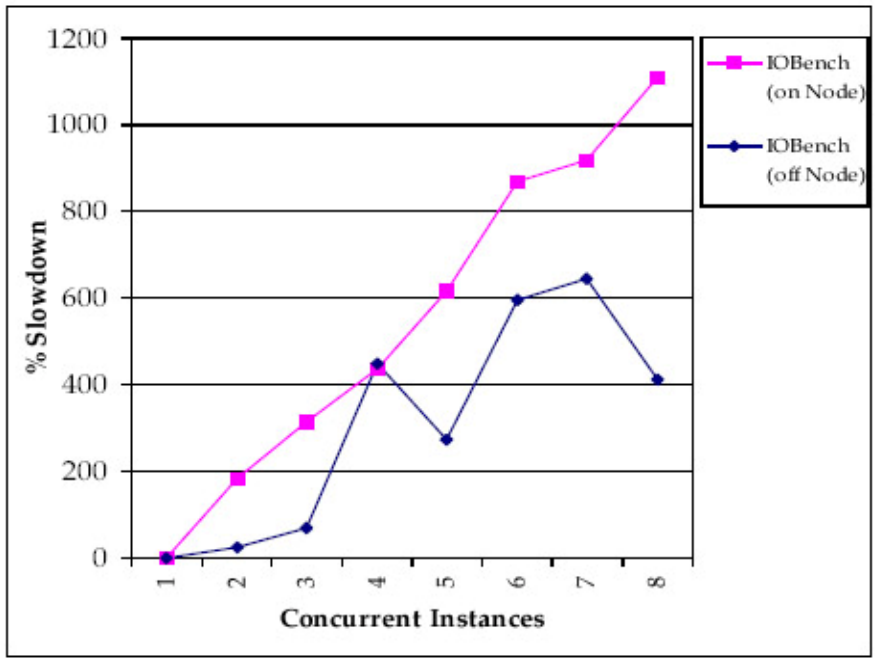
- **GUPS:** Giga-Updates-Per-Second measures the time to perform a fixed number of updates to random locations in main memory.
  (main memory)
- **STREAM:** Performs a long series of short, regularly-strided accesses through memory
  (cache)
- **I/O Bench:** Performs a series of sequential, backward, and random read and write tests
  (I/O)
- **EP:** Embarrassingly Parallel, one of the NAS Parallel Benchmarks is a compute-bound code.
  (CPU)

Resource Sharing: Effects

Memory



I/O



APP	BT	MG	FT	DT	SP	LU	CG	IS
4P	8	15	1	20	20	16	14	14
8P	12	48	30	38	33	41	54	58

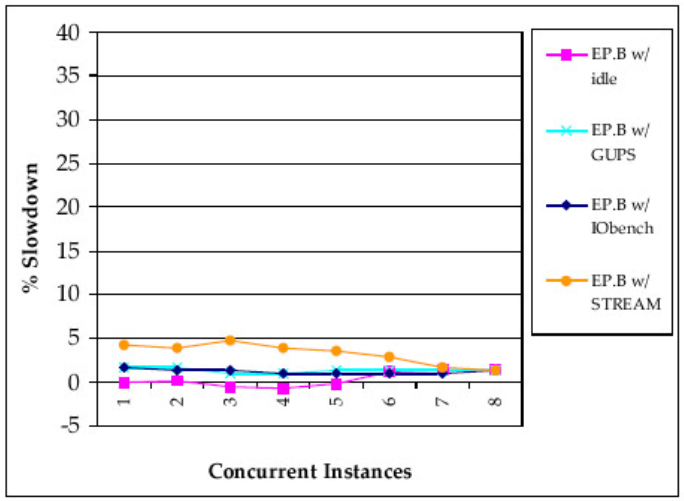
Resource Sharing: Conclusions

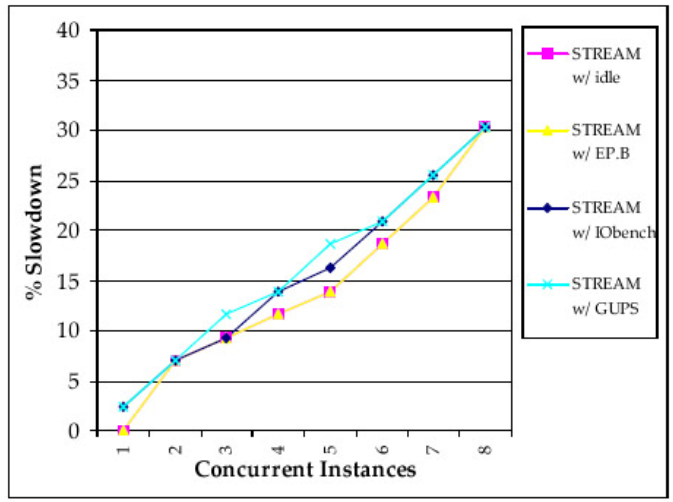
- To what extent and why do jobs interfere with themselves and each other?
  - **10-60% for memory**
  - **Super-linear for I/O**

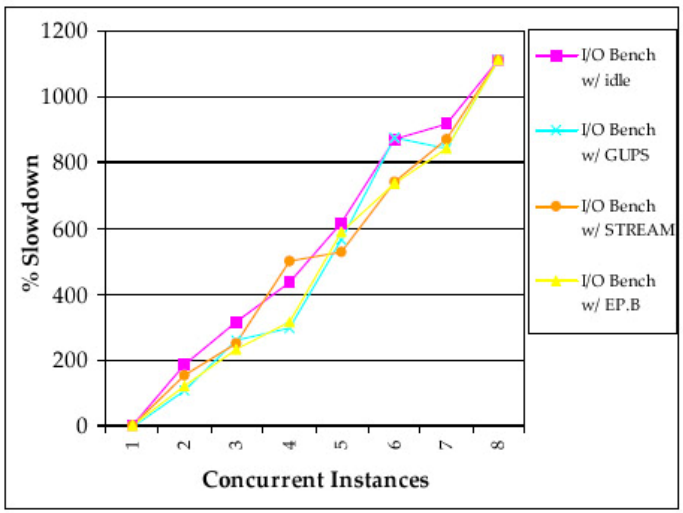
Can Symbiotic Space-Sharing Work?

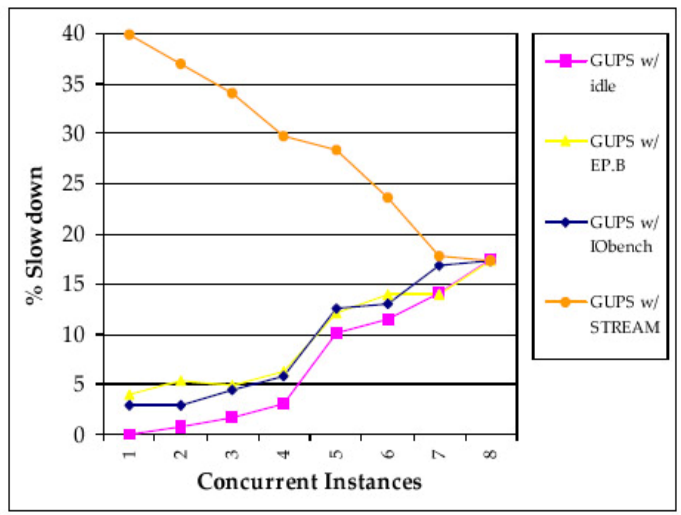
- To what extent and why do jobs interfere with themselves and each other?
- *If this interference exists, how effectively can it be reduced by alternative job mixes?*
- Are these alternative job mixes feasible for parallel codes and what is the net gain?
- How can a job scheduler create symbiotic schedules?

Mixing Jobs: Effects









Mixing Jobs: Effects on NPB

	Background Benchmark								
	BT	MG	FT	SP	LU	CG	IS	EP	I/O
BT	**12**	21	20	16	17	12	12	1	5
MG	11	**48**	25	25	25	11	11	1	4
FT	6	31	**30**	15	18	15	12	1	1
SP	21	48	36	**33**	31	23	19	2	5
LU	18	69	41	38	**41**	24	28	1	2
CG	26	82	64	42	55	**54**	36	3	7
IS	14	88	50	39	50	32	**58**	1	3
EP	2	4	4	4	4	3	2	**1**	2
I/O	-6	-2	2	-2	-6	-6	-2	-2	**1108**

- **Using NAS Benchmarks we generalize the results**
- **EP and I/O Bench are symbiotic with all**
- **Some symbiosis within the memory intensive codes**
  - CG with IS, BT with others
- **Slowdown of self is among highest observed**

Mixing Jobs: Conclusions

- Proper job mixes can mitigate slowdown from resource contention
- Applications tend to slow themselves more heavily than others
- Some symbiosis may exist even within one application category (e.g. memory-intensive)

Can Symbiotic Space-Sharing Work?

- To what extent and why do jobs interfere with themselves and each other?
- If this interference exists, how effectively can it be reduced by alternative job mixes?
- *How can parallel codes leverage this and what is the net gain?*
- How can a job scheduler create symbiotic schedules?

Parallel Jobs: Spreading Jobs

Benchmark	Speedup
BT	1.13
MG	1.34
FT	1.27
LU	1.47
CG	1.55
IS	1.12
EP	1.00
BTIO EP	1.16
BTIO SIMPLE	4.97
BTIO FULL	1.16

Speedup when 16p benchmarks are spread across 4 nodes instead of 2

Parallel Jobs: Mixing Spread Jobs

- Choose some seemingly symbiotic combinations
- Maintain speedup even with no idle processors
- CG slows down when run with BTIO(S)…

Bench A	Bench B	Speedup A	Speedup B
CG	IS	1.18	1.17
	BT	1.05	1.04
	EP	1.36	1.03
	BTIO(E)	1.38	1.07
	BTIO(S)	.55	1.03
	BTIO(F)	1.36	1.12
IS	BT	1.04	1.03
	EP	1.07	1.03
	BTIO(E)	1.11	1.07
	BTIO(S)	1.00	2.41
	BTIO(F)	1.13	1.13

Parallel Jobs: Conclusions

- **Spreading applications is beneficial (15% avg. speedup for NAS benchmarks)**

- **Speedup can be maintained with symbiotic combinations while maintaining full utilization**

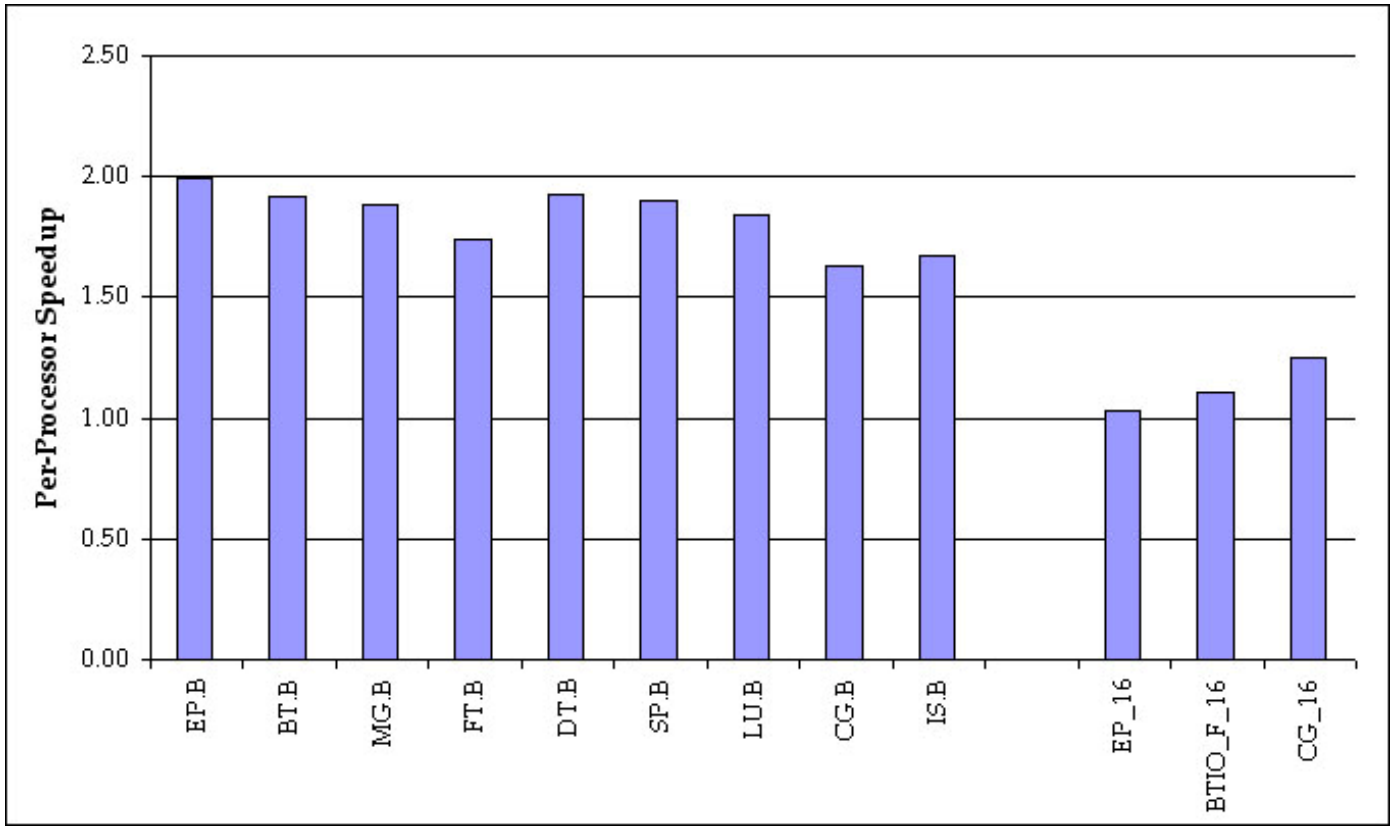
Can Symbiotic Space-Sharing Work?

- To what extent and why do jobs interfere with themselves and each other?
- If this interference exists, how effectively can it be reduced by alternative job mixes?
- How can parallel codes leverage this and what is the net gain?
- *How can a job scheduler create symbiotic schedules?*

Symbiotic Scheduler: Prototype

- **Symbiotic Scheduler vs DataStar**
- **100 randomly selected 4p and 16p jobs from:** *{IOBench.4, EP.B.4, BT.B.4, MG.B.4, FT.B.4, DT.B.4, SP.B.4, LU.B.4, CG.B.4, IS.B.4, CG.C.16, IS.C.16, EP.C.16, BTIO FULL.C.16}*
- **small jobs to large jobs: 4:3**
- **memory-intensive to compute and I/O: 2:1:1**
- **Expected runtimes were supplied to allow backfilling**
- **Symbiotic scheduler used simplistic heuristic: only schedule memory apps with compute and I/0**
- **DataStar=**5355s, **Symbiotic=**4451s, **Speedup=**1.2

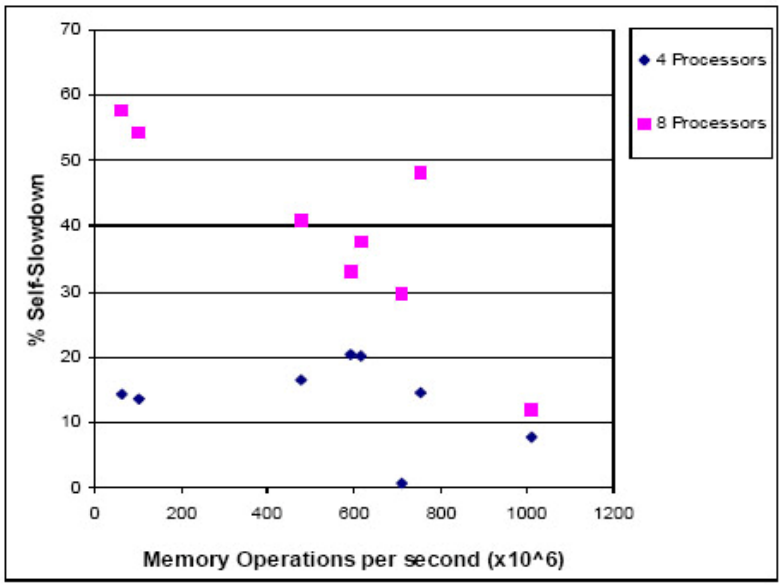
Symbiotic Scheduler: Prototype Results



- **Per-Processor Speedups (based on Avg. runtimes in test)**
  - 16-Processor Apps: 10-25% speedup
  - 4-Processor Apps: 4-20% slowdown (but double utilization)

Identifying Symbiosis

- **Ask the users**
  - Coarse Grained
  - Fine Grained
- **Online discovery**
  - Sampling (e.g. Snavely w/ SMT)
  - Profiling (e.g. Antonopoulos, Koukis w/ hw counters)



Memory operations/s vs self-slowdown

User Guidance: Why Ask Users?

- **Consent**
  - Financial
  - Technical
  - Transparency
- **Familiarity**
  - Submission flags from users are standard

User Guidance: Coarse Grained

Can users identify the resource bottlenecks of applications?

Application Workload

Applications deemed “of strategic importance to the United States federal government” by a recent $30M NSF procurement*

- **WRF**
  Weather Research Forecasting System from the DoD’s HPCMP program
- **OOCORE**
  Out Of Core solver from the DoD’s HPCMP program
- **MILC**
  MIMD Lattice Computation from the DoE’s National Energy Research Scientific Computing (NERSC) program
- **PARATEC**
  Parallel Total Energy Code from NERSC
- **HOMME**
  High Order Methods Modeling Environment from the National Center for Atmospheric Research

* High Performance Computing Systems Acquisition: Towards a Petascale Computing Environment for Science and Engineering

Expert User Inputs

App.	FLOP	Mem	I/O	Comm. Latency	Comm. Bandwidth
MILC	x			x	
PARATEC	x				x
HOMME		x			x
WRF		x			x
OOCORE			x		

- User inputs collected independently from five expert users
- Users reported to have used MPI Trace, HPMCOUNT, etc

- Are these inputs accurate enough to inform a scheduler?

User-Guided Symbiotic Schedules

	Background Benchmark						
	HOMME	WRF	BTIO	OOCORE	MILC	PARATEC	Idle
HOMME	1.00	**1.07**	*1.02*	*1.00*	*1.03*	*1.07*	*1.18*
WRF	**0.68**	1.00	*1.04*	*0.95*	*1.00*	*1.04*	*1.22*
BTIO	*1.13*	*1.18*	1.00	**.90**	*1.16*	*1.20*	*1.32*
OOCORE	*1.05*	*1.25*	**0.70**	1.00	*1.05*	*1.42*	*1.72*
MILC	1.10	1.18	1.11	1.65	1.00	1.21	*1.95*
PARATEC	1.35	1.06	1.20	1.29	1.15	1.00	*2.34*

- **The Table:**
  - 64p runs using 32-way, p690 nodes
  - Speedups are vs 2 nodes
  - Predicted Slowdown | Predicted Speedup | No Prediction
- **All applications speed up when spread (even with communication bottlenecks)**
- **Users identified non-symbiotic pairs**
- **User speedup predictions were 94% accurate**
- **Avg. speedup is 15% (Min=7%, Max=22%)**

User Guidance: Fine Grained

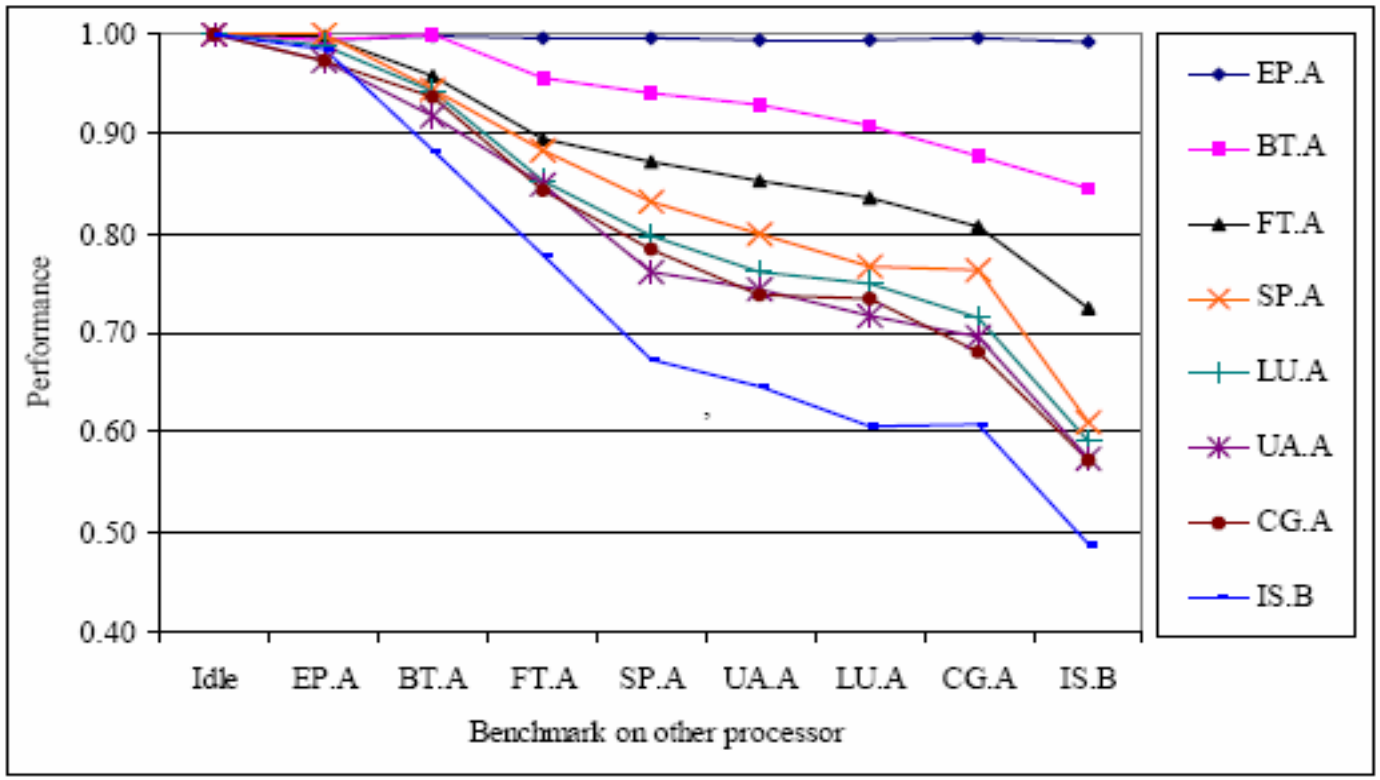
- **Submit quantitative job characterizations**
- **Scheduler *learns* good combinations on system**
- **Chameleon Framework**
  - Concise, quantitative description of application memory behavior (*signature)*
  - Tools for fast signature extraction (~5x)
  - Synthetic address traces
  - Fully tunable, executable benchmark

Chameleon: Application Signatures

	CG	FT	IS	MG	BT	LU	SP	UA
CG	–	0.13	0.14	0.14	0.15	0.12	0.13	0.13
FT	0.13	–	0.16	0.12	0.08	0.08	0.07	0.07
IS	0.14	0.16	–	0.24	0.22	0.20	0.19	0.20
MG	0.14	0.12	0.24	–	0.06	0.05	0.05	0.06
BT	0.15	0.08	0.22	0.06	–	**0.03**	**0.03**	**0.03**
LU	0.12	0.08	0.20	0.05	**0.03**	–	**0.02**	**0.03**
SP	0.13	0.07	0.19	0.05	**0.03**	**0.02**	–	**0.02**
UA	0.13	0.07	0.20	0.06	**0.03**	**0.03**	**0.02**	–

Similarity between NPB on 68 LRU Caches

Space-Sharing (Bus)



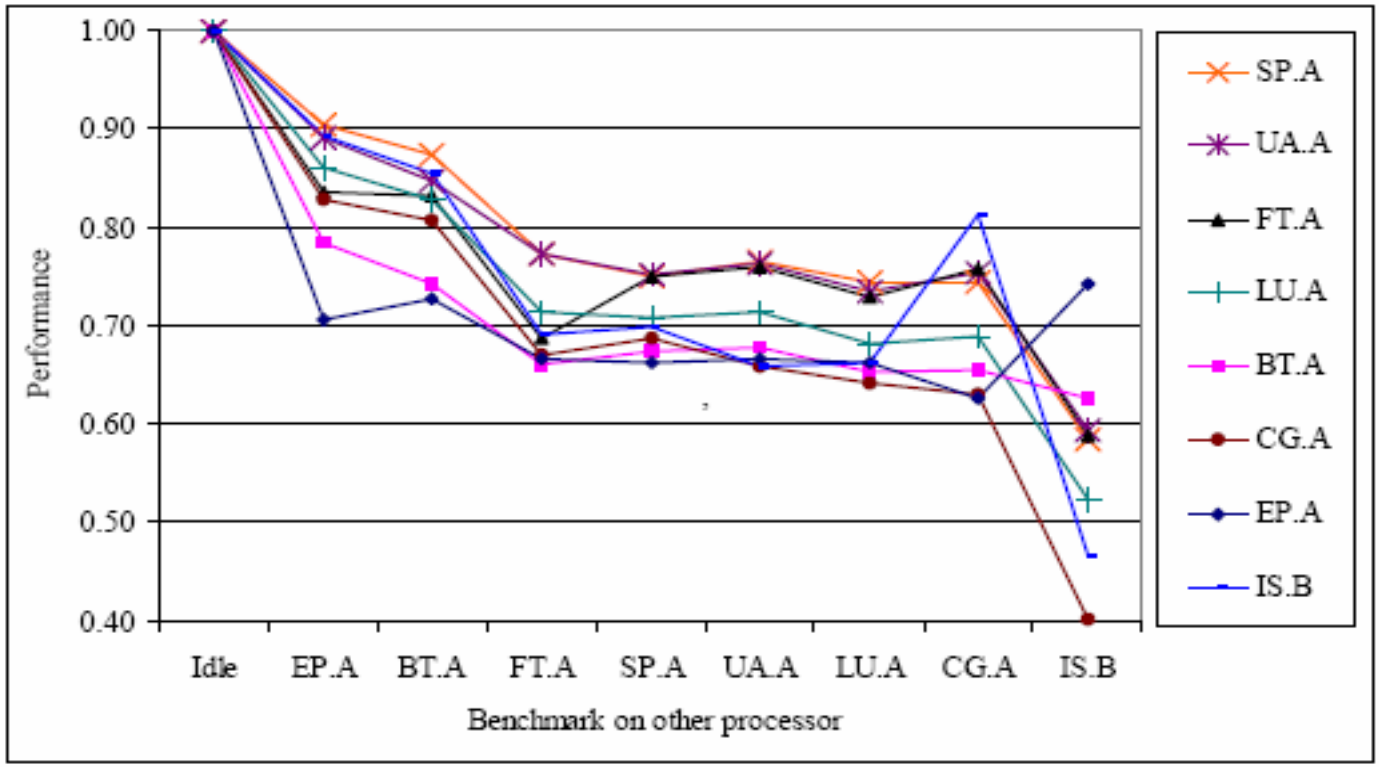
Space-sharing on the Pentium D

Comparative Performance of NPB

App	Pentium D820		Intel Centrino		IBM Power4	
	L1/L2	Performance	L1/L2	Performance	L2/L3	Performance
CG.A	.66/.99	9.69	NA	5.22	.96/1.00	3.18
FT.A	.86/.97	7.39	NA	4.75	.98/.32	9.63
MG.A	.93/.98	13.9	NA	8.37	.94/.45	5.52
IS.B	.67/.85	2.07	NA	1.68	.78/.89	.11
BT.A	.96/.98	12.1	NA	7.59	.94/.96	13.0
LU.A	.94/.95	8.25	NA	5.05	.87/.94	6.64
SP.A	.93/.94	9.06	NA	5.59	.89/.93	7.94
UA.A	.91/.91	9.19	NA	5.86	.87/.93	8.13

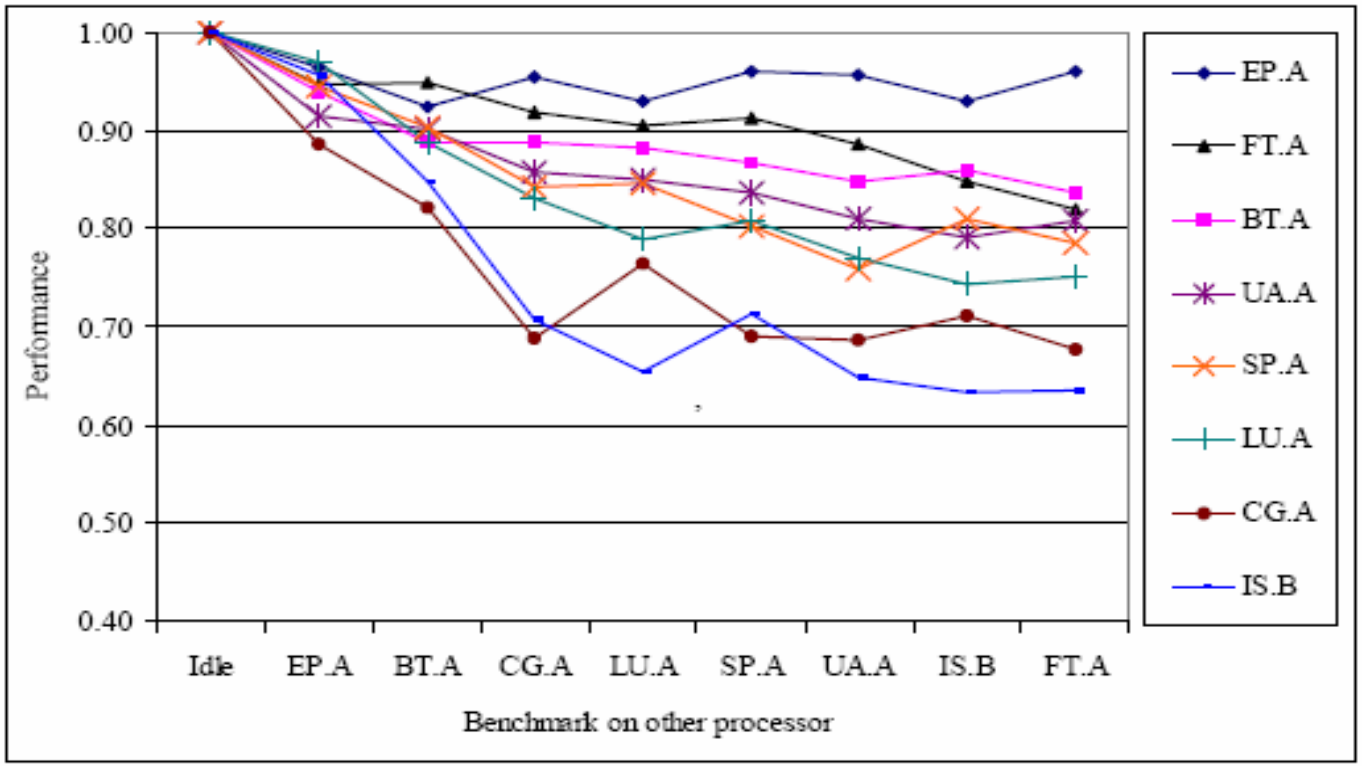
Performance in 100M memory ops per second

Space-Sharing (Bus, L2)



Space-sharing on the Intel Centrino

Space-Sharing (Bus, L2, L3)



Space-sharing on the Power4

Conclusions

- To what extent and why do jobs interfere with themselves and each other?

  10-60% for memory and 1000%+ for I/O (DataStar)

- If this interference exists, how effectively can it be reduced by alternative job mixes?

  Almost completely given the right job

- How can parallel codes leverage this and what is the net gain?

  Spread across more nodes. Normally up to 40% with our test set.

- How can a job scheduler create symbiotic schedules?

  Ask users, use hardware counters, and do future work…

Future Work

- **Workload study: How much opportunity in production workloads?**
- **Runtime symbiosis detection**
- **Scheduler Heuristics**
  - How should the scheduler actually operate?
  - Learning algorithms?
  - How will it affect fairness or other policy objectives?
- **Other Deployment Contexts:**
  - Desktop grids
  - Web servers
  - Desktops?

Thank You!