Communication with Threads in Software DSMs

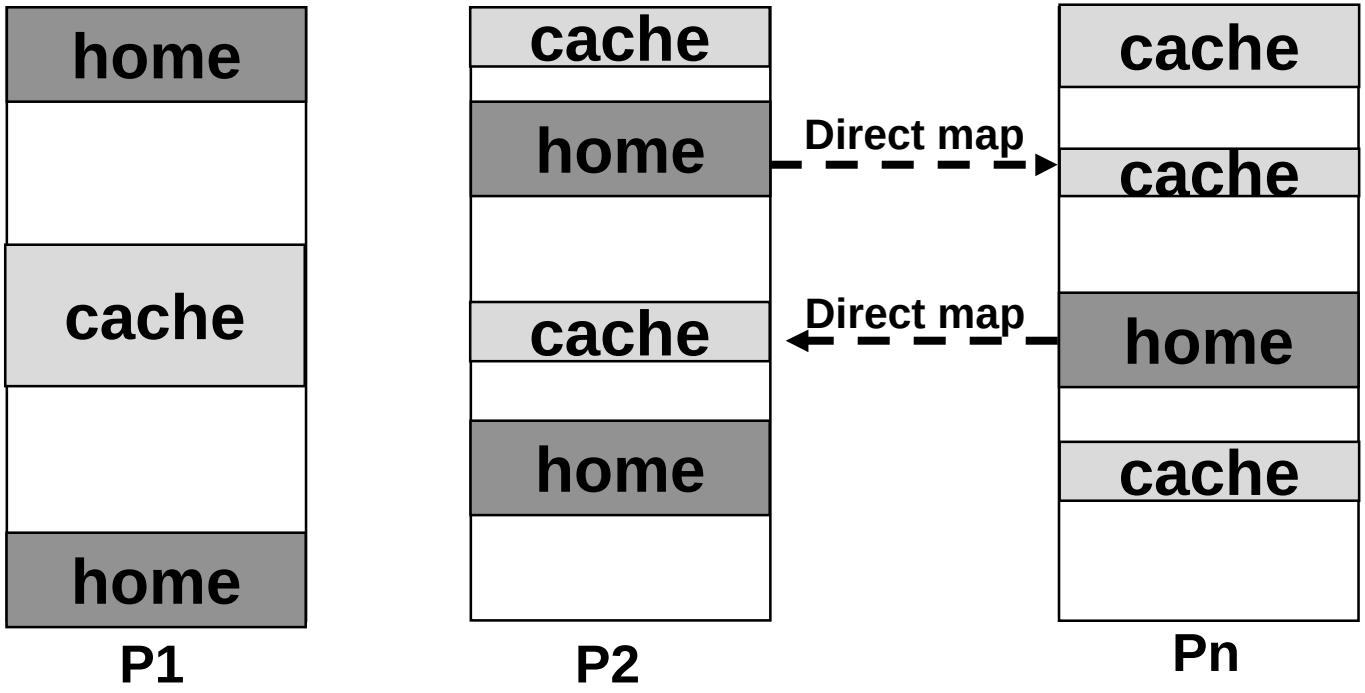
Weiwu Hu, Gang Shi, and Fuxin Zhang
Institute of Computing Technology
Chinese Academy of Sciences
E-mail:{hww,shigang,fxzhang}@ict.ac.cn

- Introduction of JIAJIA Software DSM System
- Communication Mechanism in Software DSMs
- Communication with Threads in JIAJIA
- Performance Evaluation
- Conclusion

Introduction of JIAJIA Software DSM System

- Memory management mechanism
  - Support large memory space
  - No address translation
  - CC-NUMA style
- Lock based cache coherence protocol
  - Scope Consistency
  - Lock based protocol
  - Optimization methods of protocol

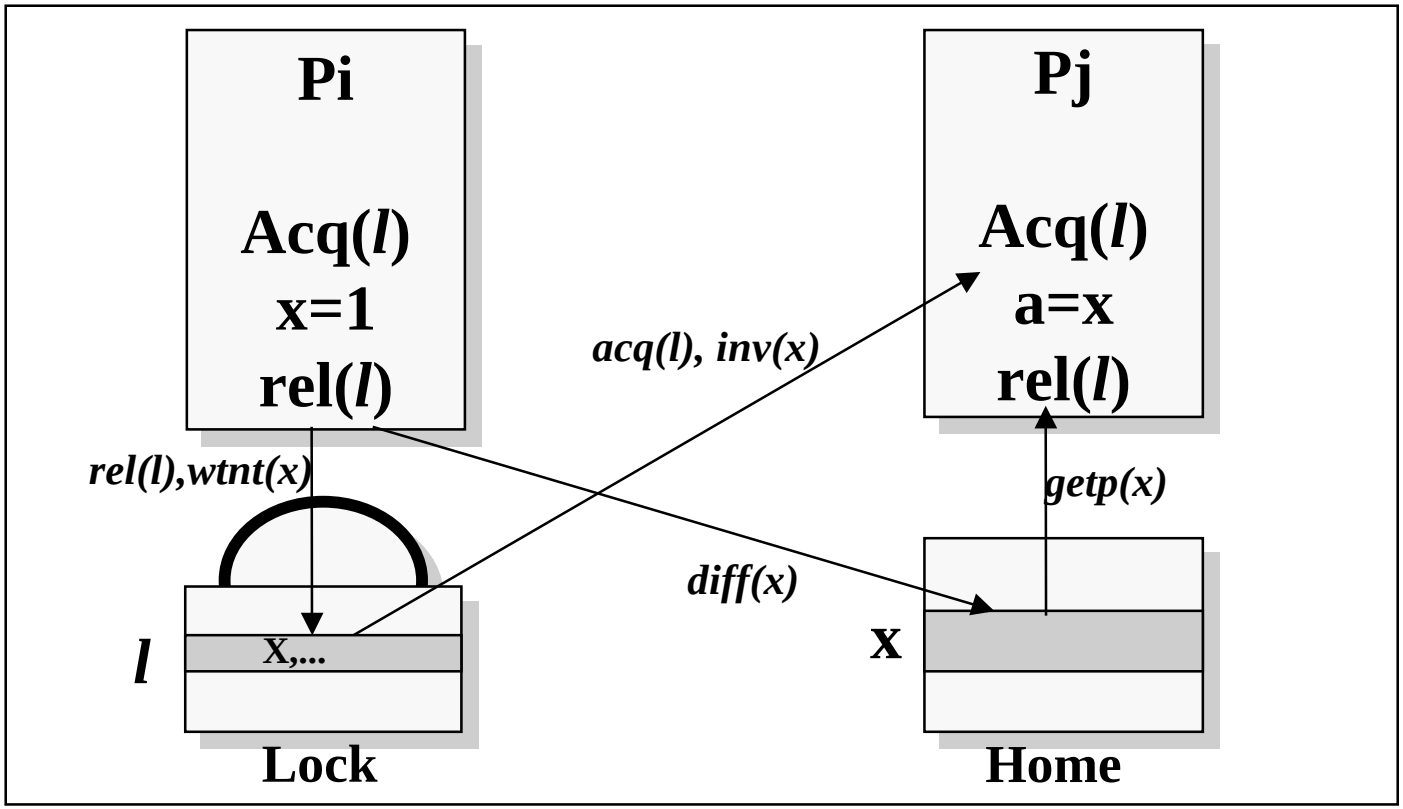
Shared Memory Organization



Support Large Shared Memory

and No Address Translation !

Lock-based Cache Coherence Protocol



Lock Operation + Write Notices ---> Scope Consistency

Many Optimization methods implemented in JIAJIA

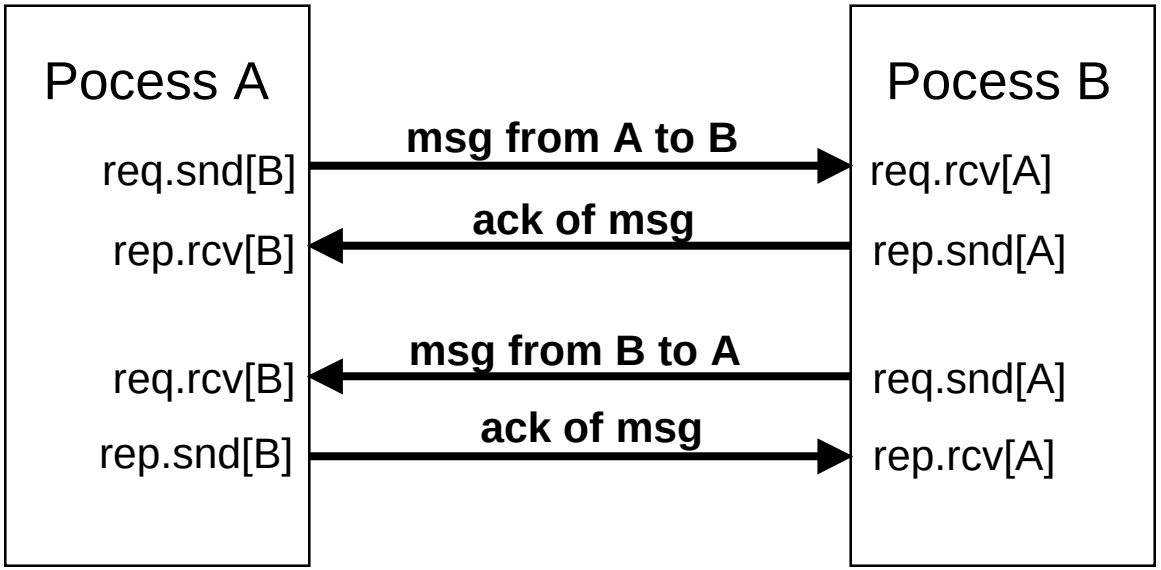
- Lazy home page write detection
- Home migration
- Write vector technique
- SMP optimization
- Dynamic data prefetching

Improve Performance Significantly!

Communication Mechanism in Software DSMs

- **Interrupt mechanism is widely used to asynchronously notify applications that messages have arrived**
  - **Merit**
    - » Easy to implement complex coherence protocols
  - **Demerit**
    - » Expensive cost **(Is it true?)**

Inter-process Communication in JIAJIA



- **For reliable message delivery over UDP, ACK mechanism was used**
- **Messages were processed in “first come first served” way**

Communication with Threads in JIAJIA

- Three communication methods with threads:
  - only one communication thread --- JIA(thr 1)
    - » responsible for sending, receiving and serving message
  - one sending thread + one receiving thread --- JIA(thr 2)
    - » responsible for sending (sending thread)
    - » responsible for receiving and serving message (receiving thread)
  - one sending thread + one receiving thread + serving thread --- JIA(thr 3)
    - » responsible for sending (sending thread)
    - » responsible for receiving (receiving thread)
    - » responsible for serving (serving thread)

Test Environment

- **PC Cluster: eight nodes, each with two 700MHz Pentium III processors and 1GB memory, 100Mbps switched Ethernet, Linux kernel 2.4**

- **PowerPC Cluster: sixteen nodes, each with a 300MHz PowerPC 604 processor and 256MB memory, AIX 4.2**

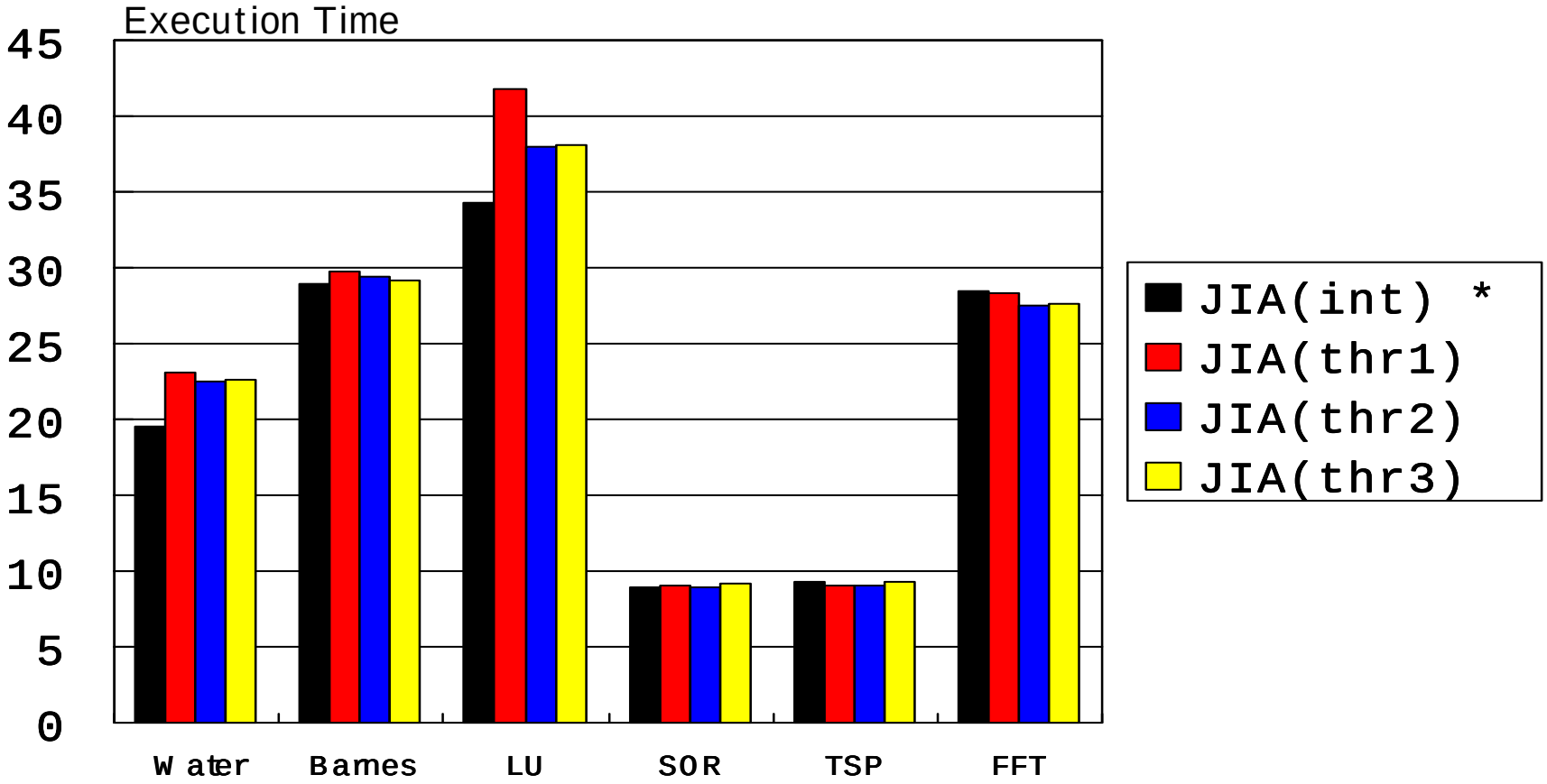
Running Results of 16-way Parallel Execution

Item		8 nodes, 16 processors, Linux						16 nodes, 16 processors, AIX					
		Water	Barnes	LU	SOR	TSP	FFT	Water	Barnes	LU	SOR	TSP	FFT
Statis-	Message(MB)	82	149	348	25	29	524	87	198	415	51	33	551
tics	GETPs	8104	17131	41579	2864	3440	61404	8686	23007	49926	6086	3918	65790
	Locks	3360	128	0	0	599	0	3360	128	0	0	607	0
	Barrs	70	28	256	200	0	36	70	28	256	200	0	36
JIA_{int}	**Total**	**19.57**	**28.87**	**34.30**	**8.90**	**9.23**	**28.44**	**25.15**	**79.31**	**66.40**	**28.66**	**17.49**	**45.15**
	Comp.	9.74	7.92	16.97	7.78	8.24	7.12	15.00	58.08	32.47	22.51	13.00	14.47
	SEGV	4.00	15.73	8.30	0.06	0.17	4.26	1.90	6.27	13.51	0.89	0.52	15.79
	Syn.	5.37	3.83	3.83	0.79	0.46	8.20	6.13	10.34	9.72	3.64	3.12	5.24
	Server	0.46	1.39	5.20	0.27	0.36	8.86	2.12	4.62	10.70	1.62	0.85	9.34
JIA_{thr1}	**Total**	**23.12**	**29.77**	**41.81**	**9.07**	**9.04**	**28.31**	**27.68**	**91.25**	**68.88**	**29.16**	**20.93**	**71.34**
	main	10.38	8.20	21.41	8.64	8.07	7.33	16.98	60.54	38.53	25.79	13.63	18.83
	snd+rcv+serv	0.09	0.18	0.40	0.05	0.02	0.72	0.53	3.58	1.91	0.38	0.18	5.29
	wait	12.65	21.39	20.00	0.38	0.95	20.26	10.17	27.13	28.44	2.99	7.12	47.22
JIA_{thr2}	**Total**	**22.51**	**29.42**	**37.94**	**8.96**	**9.06**	**27.55**	**27.53**	**82.78**	**67.00**	**29.18**	**19.50**	**51.39**
	main	10.36	8.21	21.27	8.61	8.08	7.44	16.94	60.37	38.94	25.84	13.48	19.09
	send	0.04	0.08	0.17	0.02	0.01	0.56	0.20	0.54	0.73	0.10	0.05	3.16
	recv+server	0.08	0.11	0.23	0.04	0.02	0.34	0.35	0.83	1.45	0.32	0.14	2.34
	wait	12.03	21.02	16.27	0.29	0.95	19.21	10.04	21.04	25.88	2.92	5.83	26.80
JIA_{thr3}	**Total**	**22.65**	**29.19**	**38.05**	**9.13**	**9.29**	**27.67**	**28.10**	**82.40**	**67.25**	**29.05**	**19.74**	**52.25**
	main	10.39	8.23	21.25	8.64	8.27	7.53	17.06	60.37	38.98	25.84	13.29	19.16
	send	0.04	0.09	0.17	0.02	0.01	0.54	0.23	0.53	0.89	0.12	0.03	3.12
	recv	0.01	0.02	0.06	0.01	0.00	0.04	0.09	0.12	0.31	0.06	0.03	0.25
	server	0.07	0.15	0.21	0.03	0.02	0.39	0.29	0.69	1.08	0.29	0.11	2.15
	wait	12.14	20.70	16.36	0.43	0.99	19.17	10.43	20.69	25.99	2.74	6.28	27.57

Running Results of 8-way Parallel Execution

Item		4 nodes, 8 processors						8 nodes, 8 processors					
		Water	Barnes	LU	SOR	TSP	FFT	Water	Barnes	LU	SOR	TSP	FFT
Statistics	Message(MB)	36	70	206	11	18	439	41	81	277	24	21	510
	GETPs	3546	7787	24790	1240	2108	52428	3768	8955	33228	2862	2466	61166
	Locks	1040	64	0	0	553	0	1040	64	0	0	552	0
	Barrs	70	28	256	200	0	36	70	28	256	200	0	36
JIA_{int}	**Total**	**23.94**	**25.98**	**53.17**	**16.47**	**15.49**	**38.44**	**24.50**	**26.45**	**56.64**	**13.70**	**15.73**	**33.84**
	Comp.	19.34	15.37	38.50	15.43	14.56	14.99	19.44	14.67	37.65	12.30	14.72	11.27
	SEGV	1.25	5.84	4.66	0.04	0.19	15.25	1.79	7.85	6.40	0.10	0.29	14.55
	Syn.	2.65	3.47	5.63	0.80	0.35	2.91	2.82	2.50	7.07	0.70	0.26	1.65
	Server	0.70	1.35	4.38	0.20	0.39	5.29	0.45	1.43	5.52	0.60	0.46	6.37
JIA_{thr1}	**Total**	**25.63**	**26.23**	**54.29**	**16.34**	**15.59**	**40.45**	**24.35**	**26.55**	**49.52**	**13.35**	**15.38**	**30.08**
	main	20.41	15.73	41.53	16.25	14.81	15.11	20.41	14.91	41.07	13.14	14.78	11.90
	snd+rcv+serv	0.17	0.21	0.33	0.04	0.02	1.33	0.10	0.20	0.39	0.07	0.03	1.23
	wait	5.05	10.29	12.43	0.05	0.76	24.01	3.84	11.44	8.06	0.14	0.57	16.95
JIA_{thr2}	**Total**	**25.28**	**26.16**	**53.52**	**16.44**	**15.25**	**40.46**	**24.24**	**26.34**	**50.75**	**13.36**	**15.39**	**29.91**
	main	20.42	15.76	41.53	16.27	14.50	14.56	20.39	14.93	40.69	13.17	14.81	11.88
	send	0.05	0.12	0.14	0.02	0.02	1.05	0.05	0.10	0.12	0.02	0.01	1.07
	recv+server	0.12	0.13	0.25	0.03	0.02	0.49	0.09	0.13	0.18	0.04	0.03	0.53
	wait	4.69	10.15	11.68	0.12	0.71	24.36	3.71	11.18	9.76	0.13	0.54	16.43
JIA_{thr3}	**Total**	**25.19**	**26.07**	**54.26**	**16.62**	**15.49**	**40.97**	**24.28**	**25.93**	**51.23**	**13.49**	**15.42**	**30.23**
	main	20.45	15.81	41.70	16.34	14.56	14.84	20.46	14.97	40.77	13.13	14.84	11.82
	send	0.05	0.13	0.15	0.02	0.01	1.05	0.05	0.08	0.12	0.02	0.01	1.15
	recv	0.01	0.03	0.08	0.01	0.01	0.08	0.01	0.01	0.08	0.01	0.01	0.06
	server	0.13	0.14	0.30	0.04	0.03	0.60	0.09	0.15	0.29	0.05	0.04	0.58
	wait	4.55	9.96	12.03	0.21	0.88	24.40	3.67	10.72	9.97	0.28	0.52	16.62

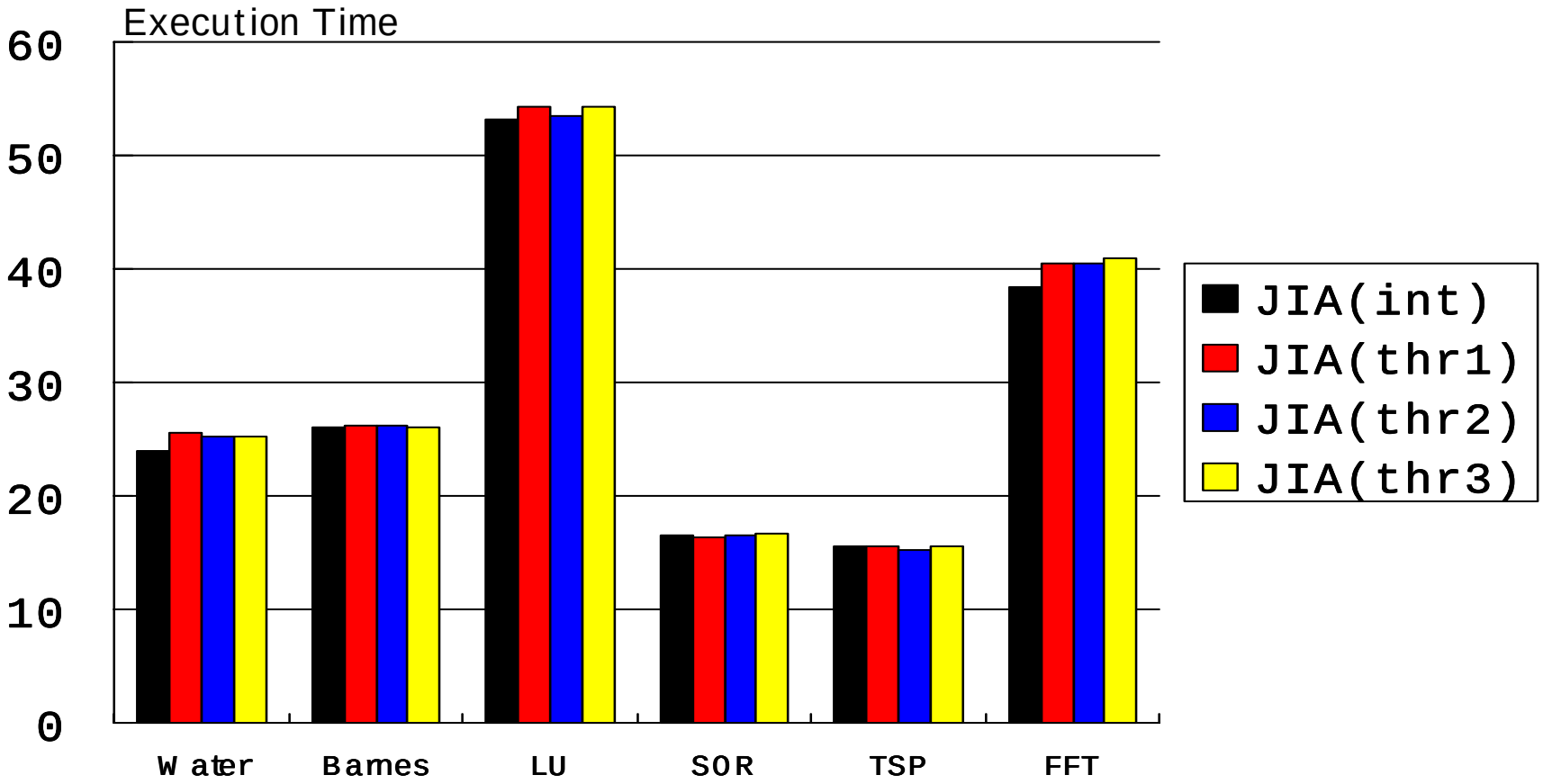
Performance Evaluation (I)



16-way Parallel execution: 8 nodes, 16processors, Linux

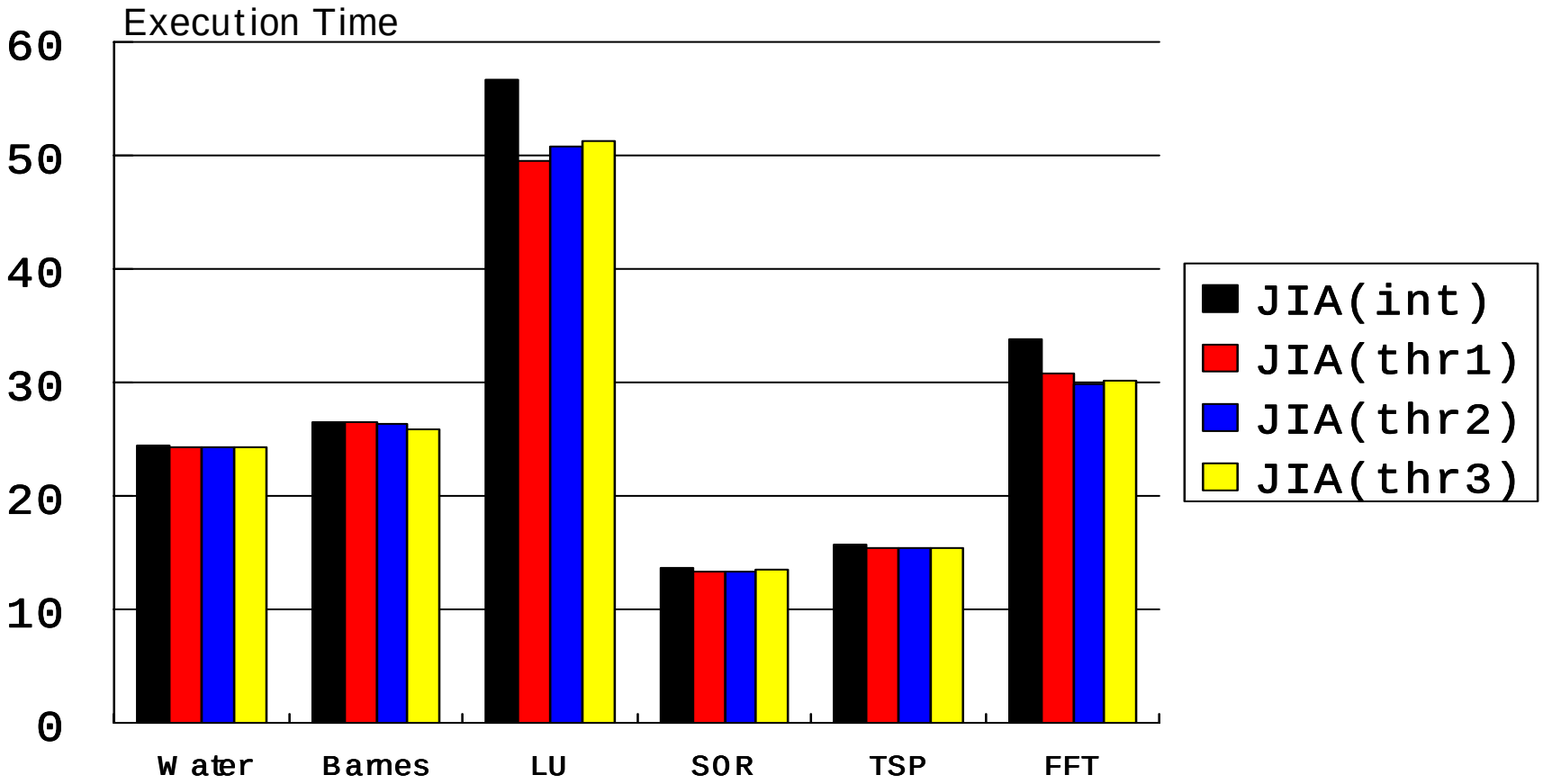
***: using interrupt mechanism.**

Performance Evaluation (I)



8-way Parallel execution: 4 nodes, 8processors, Linux

Performance Evaluation (I)

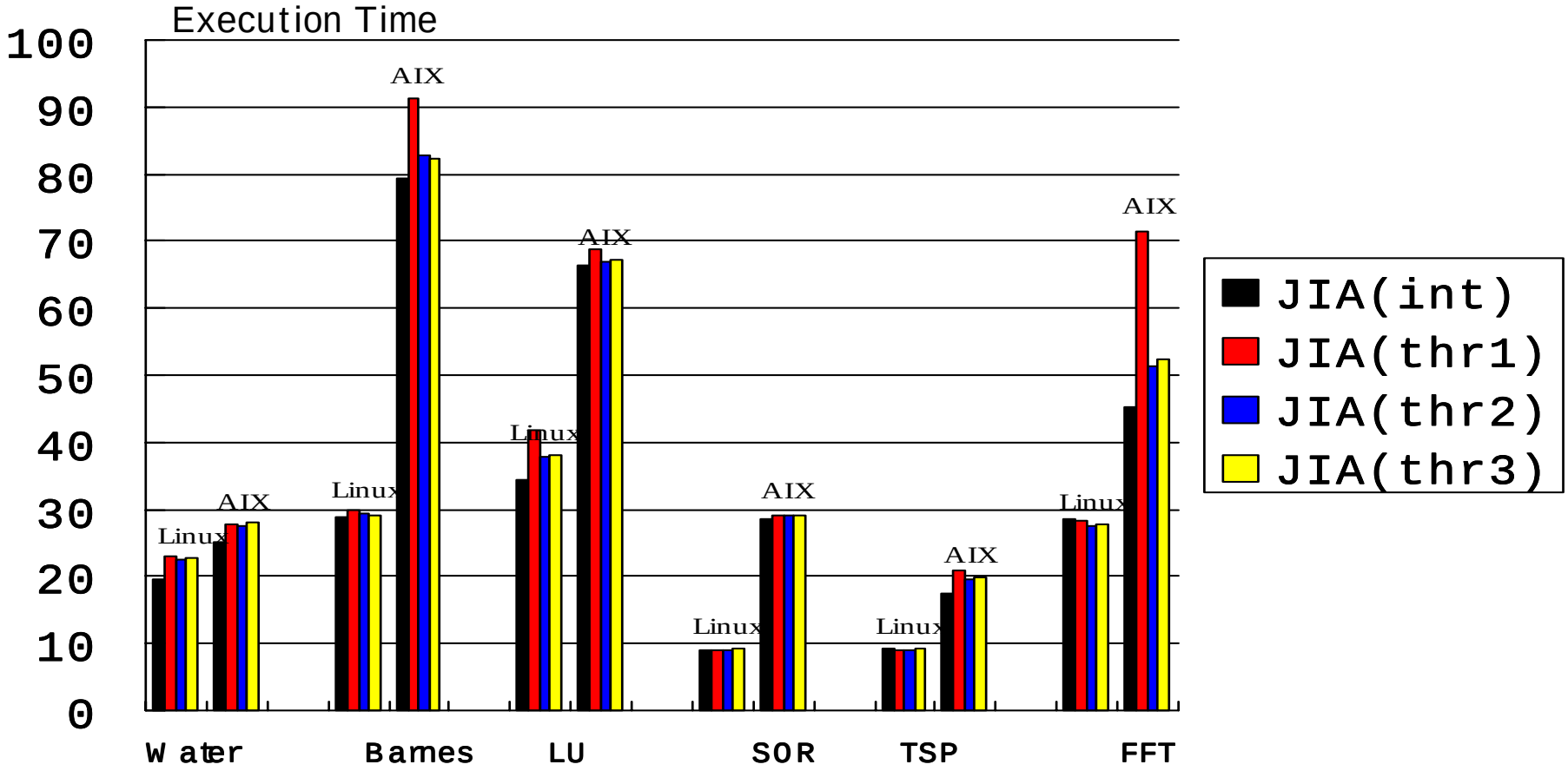


8-way Parallel execution: 8 nodes, 8processors, Linux

Conclusion of Performance Evaluation (I)

- Performance of threads communication is slightly worse than that of interrupt communication when there is only one processor to run a process (from slide 12 and 13).
- Performance of threads communication is better than that of interrupt communication when there is more than one processor in a node to run a process (from slide 14).
- The thread communication method with one sending thread and one receiving thread (JIA(thr2)) achieves the best performance in the three thread communication methods (from slide 12, 13, and 14).

Performance Evaluation (II)

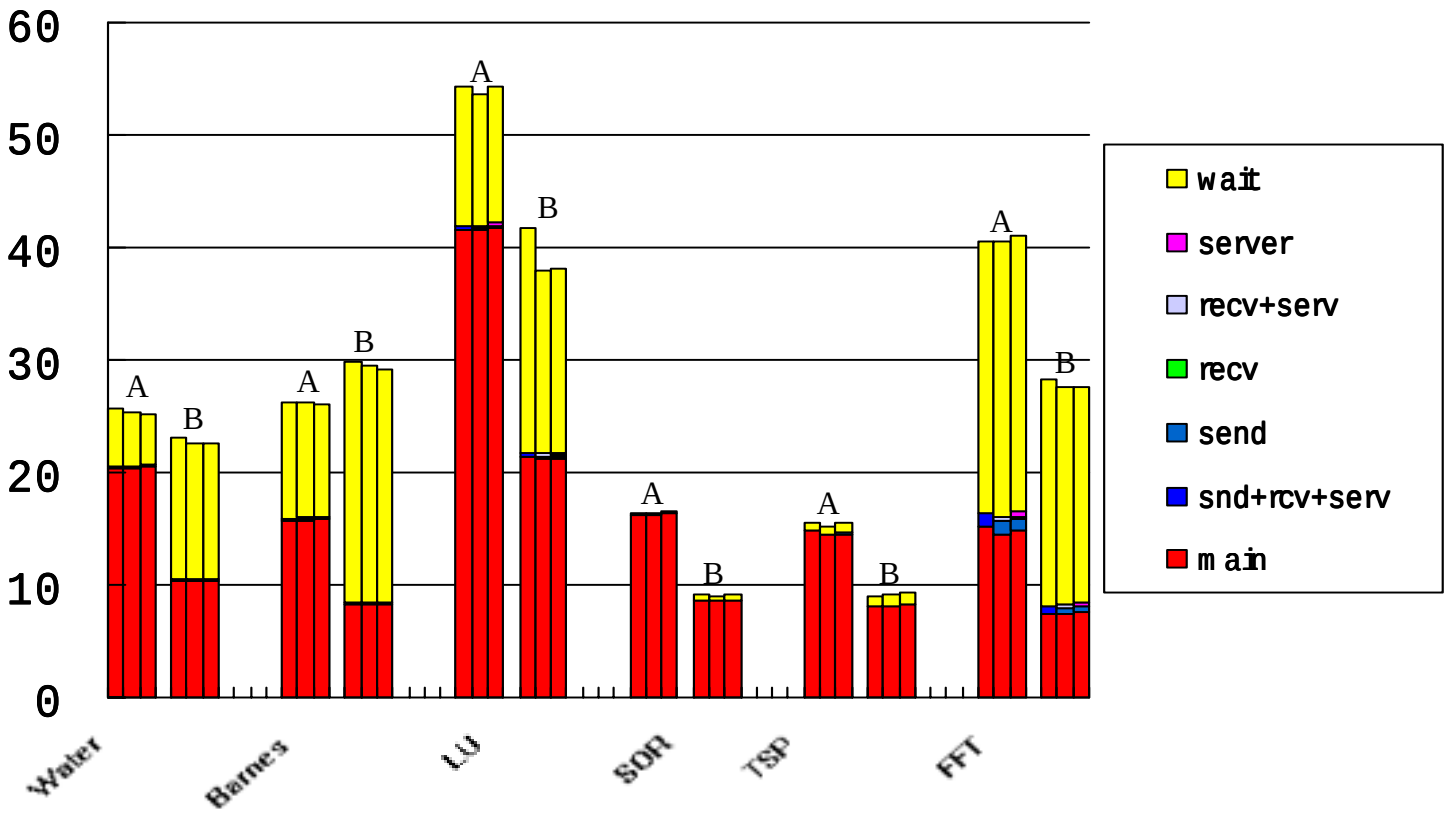


16-way Parallel execution: in Linux and AIX

Conclusion of Performance Evaluation (II)

- The effect of the thread communication is related to the operation system (from slide 16).
- In Linux, the performance of only Water an d LU varies clearly with different thread communication methods, while in AIX, communication methods affect the execution time of Water, Barnes, TSP, and FFT significantly.
- The different thread scheduling methods in Linux and AIX have different effects on the responsiveness of requests.

Performance Evaluation (III)



A: 8-way Parallel execution, 4 nodes, 8processors, Linux

B: 16-way parallel execution, 8 nodes, 16processors, Linux

Conclusion of Performance Evaluation (III)

- The run time difference of different communication methods is mainly caused by the difference of waiting time (the yellow part in slide 18).
- The time for main thread scales down when the number of processors changes from 8 to 16, but the waiting time increases significantly.
- The waiting time constitutes the major overhead of communication substrate of software DSMs. It implies that software DSMs need the quick responsiveness of requests, the prompter the better.