Study of Relative Performance Impact: Coherence Protocol vs. Network Speed [1]

Zak Smith   Mostafa Arifin

May 8, 1998

[1]Completed as semester project for CS/ECE 757, Prof. Mark Hill

Abstract

Architects and implementors must know design trade–offs to design successful systems. Two important aspects of multiprocessor performance are *network quality* and *coherence protocol*. We study the performance impact of coherence protocol choice (MSI vs. MESI) as compared to the performance impact of the high–speed network. A higher quality network is either wider, or has less latency, or both.

We find that in the majority of cases, *network quality dominates the effects of coherence protocol* — that is, lower–quality networks with the MESI protocol do not yield faster runs that higher–quality networks with the MSI protocol. We also note that program characteristics can affect which protocol performs better.

Contents

Chapter 1

Introduction

1.1 Topic Importance

In order for a machine to achieve high performance within a cost constraint, architects and implementors must know the benefits and trade-offs for different design decisions.

The performance of the interconnection network is key in multiprocessor systems. In all but "embarrassingly parallel" applications, communication between processing elements provides those elements with the data they need to continue processing.

Our study examines three parameters which affect network performance: coherence protocol, flit delay, and flit size. We believe this type of study is important to verify theoretical models with real benchmark results.

1.2 Hypothesis

During lecture, it was noted that a good implementation of the interconnection network often dominates other factors such as network topology. Since significant time was spent discussing coherence protocols [1], we decided to determine the relative performance impact of coherence protocol as compared to "quality of network implementation." Thus, our hypothesis:

A good network implementation dominates the effects of choosing a specific coherence protocol.

1.3 Definitions

1.3.1 Flit Delay

The flit delay is the time it takes for a flit to pass through a network switch, relative to the clock speed of the processor. For example, if the flit delay was 2, and the processor was running at 300 Mhz, it would take 6.6 ns for a flit to propagate through a switch. This primarily affects latency.

1.3.2 Flit Width

The flit width specifies the number of bytes in each network flit, which is equivalent to the width of the network. This primarily affects bandwidth.

1.3.3 Coherence Protocol

The two coherence protocols we look at are the MSI and MESI protocols discussed in lecture and in [1].

MESI

M	Modified	This is the only copy, dirty.
E	Exclusive	Ensured to be the only RO copy.
S	Shared	One of two or more RO copies.
I	Invalid	Block not present.

MSI

M	Modified	This is the only copy, dirty.
S	Shared	One of one or more RO copies.
I	Invalid	Block not present.

Thus the primary difference between the two is that, for MSI, if a processor reads a block known to no other processor and then writes it, it must notify the directory (S to M). For MESI, this is not the case: a processor that reads the block first has it in Exclusive and can move to Modified without any traffic to the directory (E to M).

At first glance, it would appear that MESI would perform better, although as we will see later, there can be implementation issues which make this not the case.

Chapter 2

Methods

2.1 Simulator Model

We chose to use RSIM [7,3], an ILP multiprocessor simulator developed at Rice University. The system simulates a collection of workstation–like nodes connected in a 2d mesh interconnection network, with CC-NUMA based on a distributed directory.

We simulated a 16 processor system to control runtime of our simulations.

2.2 Search-space

One of the parameters we want to change is "quality of network implementation." We decided this is basically a combination of flit delay (related to the clock speed of the network), and the flit size (width of the network). In other words, the "best" network will have the fastest clock speed and the widest data-path. The "worst" network will have the slowest clock speed and the the narrowest data-path.

To cover a range of network qualities, we chose to use flit widths of 2, 4, 8 bytes, and flit delays of 2, 4, 8 processor clocks. Thus for each benchmark, we have 18 data-points:

	FD2	FD4	FD8
FS8	MESI, MSI	MESI, MSI	MESI, MSI
FS4	MESI, MSI	MESI, MSI	MESI, MSI
FS2	MESI, MSI	MESI, MSI	MESI, MSI

2.3 Benchmarks

There are 5 benchmark programs available for RSIM from the SPLASH [5] and SPLASH-2 [6] sets: mp3d, fft, radix, lu, and water. These will be described in the results section.

2.4 Data–sets

Condor [4], is a system for high–throughput computing available here at the UW. It is a great tool for architects because simulations typically take days to finish. Condor allows many of these jobs to be distributed to idle machines. Normal condor jobs are check-pointed and "migrated" off machines when a user logs on or the machine is to be powered off. This allows the job continue where it left off when another machine becomes available.

Because of some tricky code in RSIM, it would not work correctly when linked against the Condor libraries. Thus we were limited to running RSIM as a "vanilla" job — that is, when the job is interrupted by a user or the machine being powered down, instead of check-pointing the job so it could continue on another machine, the vanilla jobs are killed and must be restarted. Because of this, we had to make sure our simulations would finish in less than about 12 hours! This limited the number of processors we could simulate and the size of the data-sets run in each application.

We have no results for water because none of the nontrivial data-sets could finish in the average idle time of a machine in the Condor pool.

Chapter 3

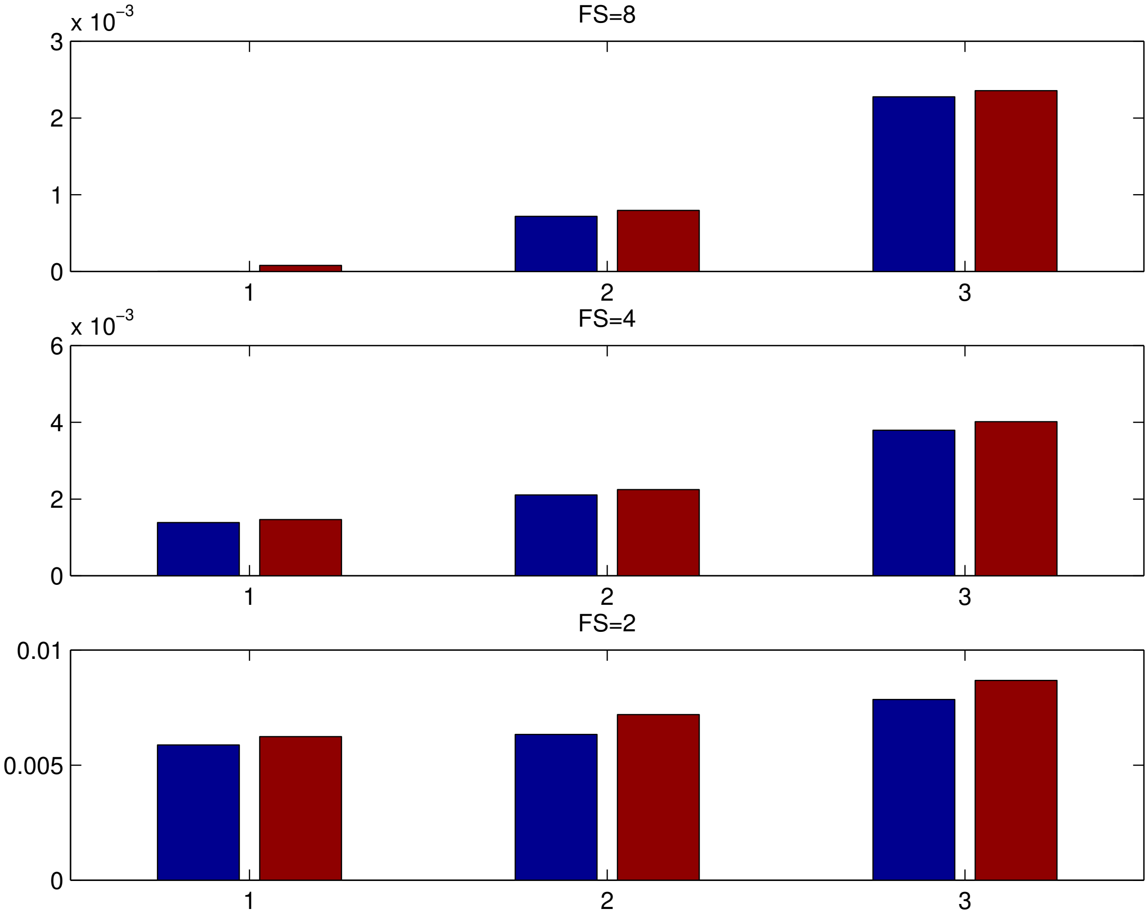
Results

3.1 FFT

3.1.1 Description

FFT is a complex 1-D version of the radix-$\sqrt{n}$ six-step algorithm described by David Bailey. It is optimized to minimize interprocessor communication. [6]

3.1.2 Normalized Runtime



Description of graph: The first graph (first row) represents all the data for flit size = 8, the second row represents flit size = 4, and the bottom row represents flit size = 2. In each graph the first pair of data (1) is flit delay = 2, (2) is flit delay = 4, and the last (3) is flit delay = 8.

The first of each pair (blue) represents the MESI runtime, the second (red) represents the MSI runtime. The magnitude of the graph is the performance degradation relative to the baseline value (FS8, FD2, MESI). Thus we can see the slowest run was FS2/FD8, the value in the lower right–hand corner. The fastest run was FS8/FD2, the value in the upper left–hand corner.

It is also important to note that for any pair, neighboring values (those above, below, left, or right one unit) differ by only one "step" of FS (vertical hops) or FD (horizontal hops).

Using this format of graph, it is possible to see the dominance of network quality (FS and FD) over protocol choice: there is no case such that the MESI run of a lower quality network is faster than the MSI run for a higher quality network. It is easier to visualize this relation using a neighbor dominance graph.

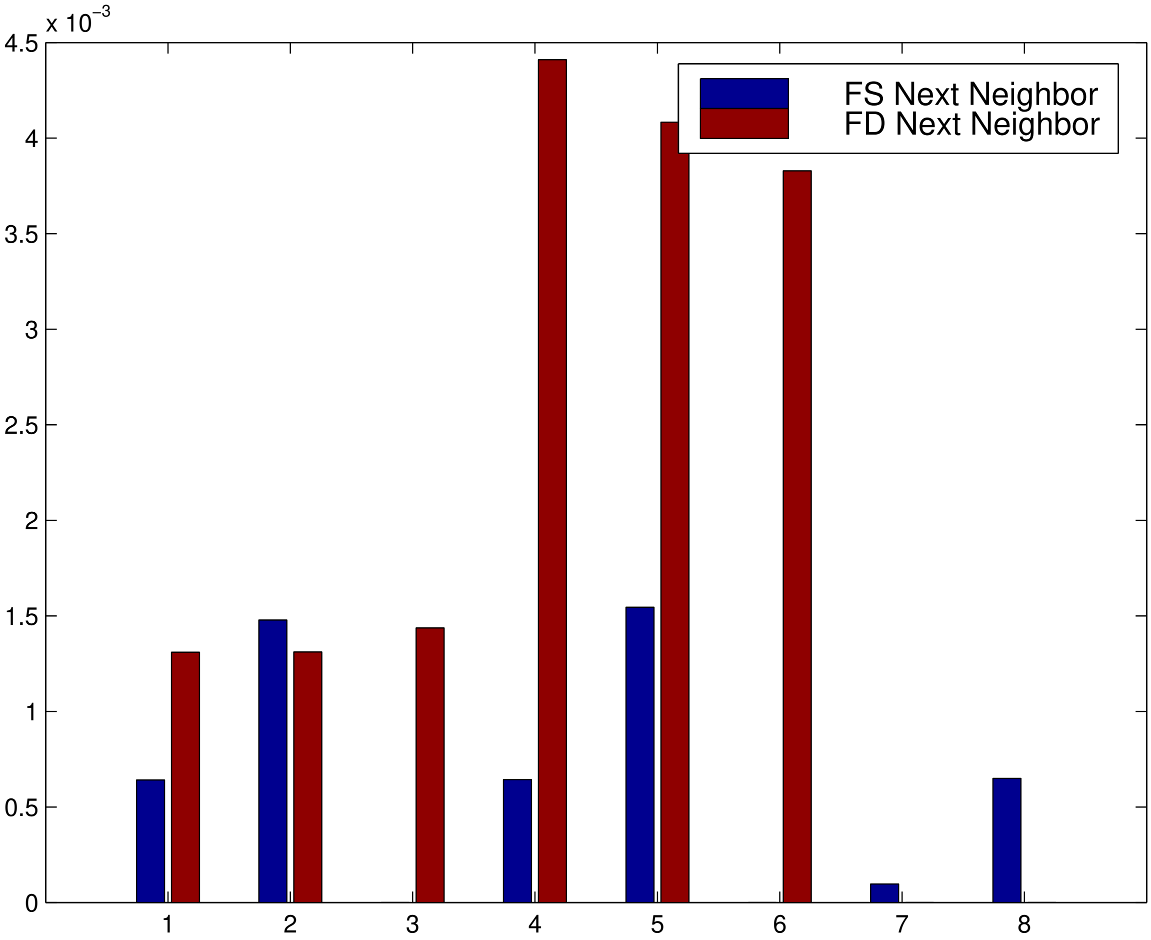
3.1.3 Runtime Discussion

We can see that, for FFT, the protocol choice has little effect on runtime when compared to flit delay or flit size. Flit size has the largest impact, that is, all the runs with FS2 are much slower than those with FS4, and similarly for FS4 over FS2.

It is also important to note that in all cases, the MESI protocol run is faster than the MSI protocol run.

It is also important to note the scale of the runtime slowdown: the slowest run is less than 1% slower than the fastest run. This is probably because this FFT implementation is optimized to reduce communication, handles latency well, and does not have lots of communication on the critical path.

3.1.4 Neighbor Dominance



The neighbor dominance graph shows the *minimum* difference in performance between each network quality and all of its "one–hop" neighbors which have a lower quality network. The blue bar of each pair is the common–flit–size neighbor (flit delays differ), and the red bar of each pair is the common–flit–delay neighbor (flit sizes differ). A positive value means the lower quality network always had a slower runtime. Each bar is computed: diff = min(worse quality network's MSI, MESI) - max(better quality network's MSI, MESI).

Pair 3 and pair 6 have no common–flit–size lower–quality neighbors because they correspond to the FS8/FD8 and FS4/FD8 runs — we tested no flit delays > 8. Similarly, pair 7 and pair 8 have no common–flit–delay neighbors because we tested no flit sizes < 2. There is no pair 9 because FS2/FD8 is the worst network quality we tested.

For example, pair 1 indicates two things: FS8/FD4/MSI is 0.0641% slower than FS8/FD2/MESI, and FS4/FD2/MSI is 0.131% slower than FS8/FD2/MESI.

Positive values indicate our hypothesis is supported: the network quality *does* dominate protocol effect. Negative values, if any, indicate that a lower–quality network's MSI run is faster than a faster network's MESI run (or vice versa).

3.1.5 Dominance Discussion

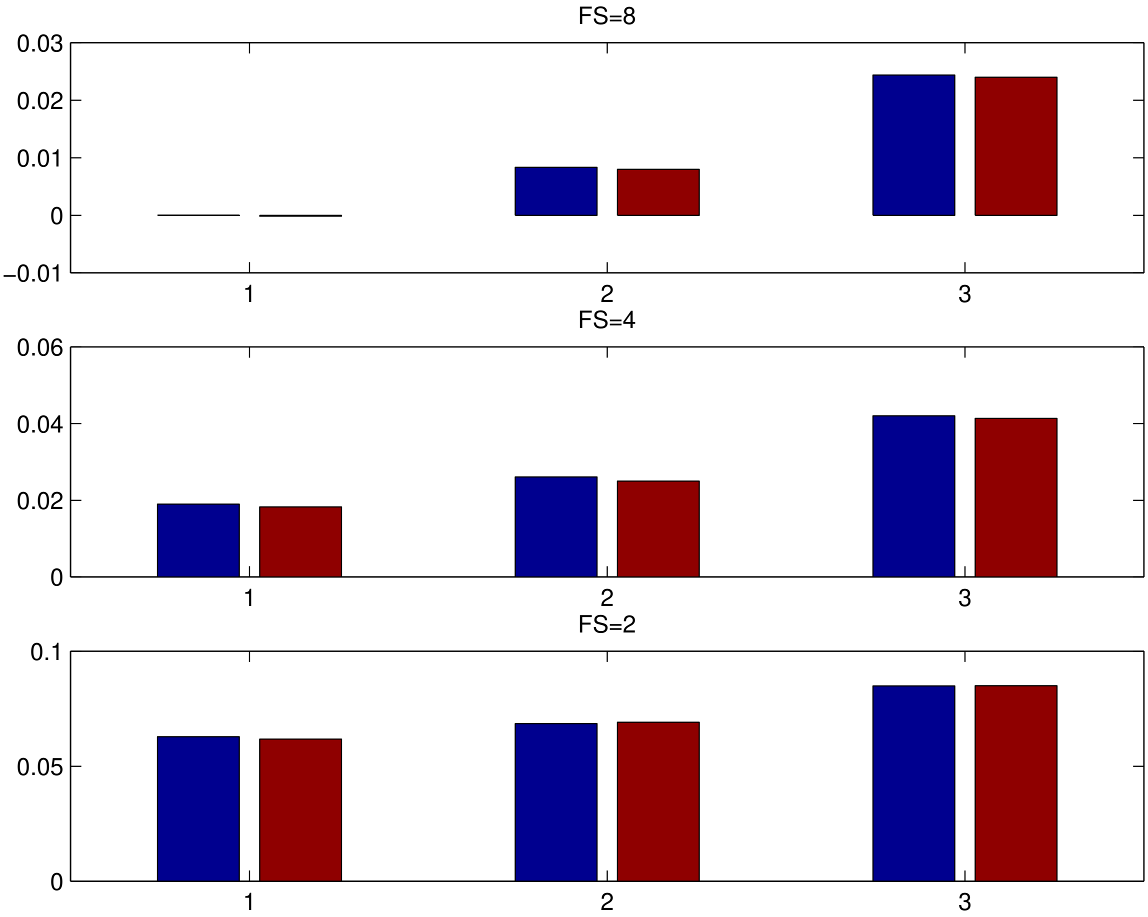
For FFT, it is important to note that our hypothesis is supported in all cases. Also, from the common–flit–delay bars (red), we can see decreasing the flit size has a larger negative effect than increasing the flit delay. In other words, it's easer to make the program speed up with a wider network than by decreasing the flit delay.

3.2 MP3D

3.2.1 Description

MP3D simulates the flow of a rarefied fluid, for example, around a spacecraft as it enters the atmosphere. Each molecule is statically scheduled on a processor, and data sharing occurs when particles collide. [5]

3.2.2 Normalized Runtime



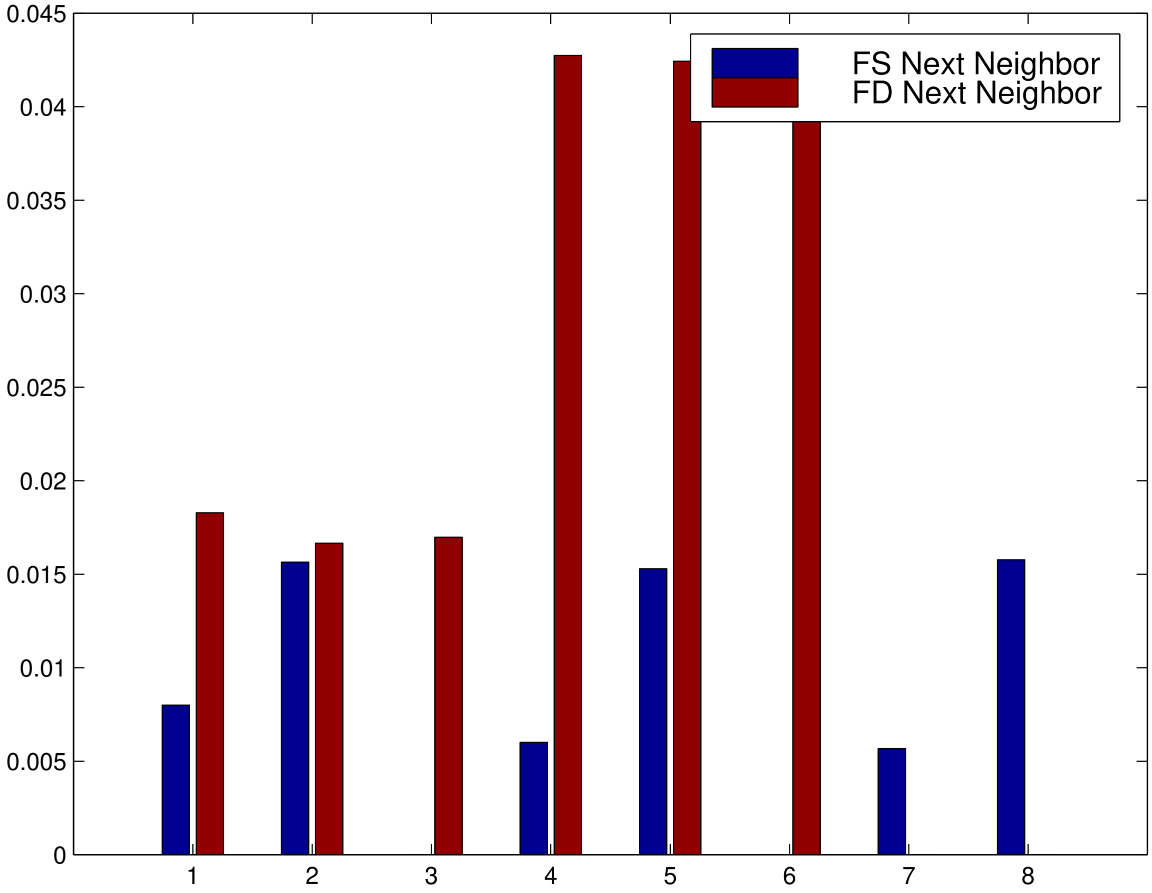
3.2.3 Runtime Discussion

Similar to FFT, the protocol choice has little effect on the runtime when compared to flit delay or flit size. Flit size again has the largest impact, that is, all the runs with FS2 are much slower than those with FS4, and similarly for FS4 over FS2.

In MP3D, however, the MESI runs are not always faster than the MSI runs. In fact, the "baseline" run was not the fastest run in this set — that's why the first graph not based at 0! MSI will perform better than MESI in the RSIM implementation when it is often the case that EX blocks are replaced in a processor's cache. It must notify the directory. Another case in which MSI will outperform MESI is when the first read to a block (P1 has it in E) is followed by many other processors' reads of the same block (P1 must move to S).

The scale of runtime slowdown is much greater than that seen for FFT. The slowest MP3D run is 9% slower than its fastest run.

3.2.4 Neighbor Dominance



3.2.5 Dominance Discussion

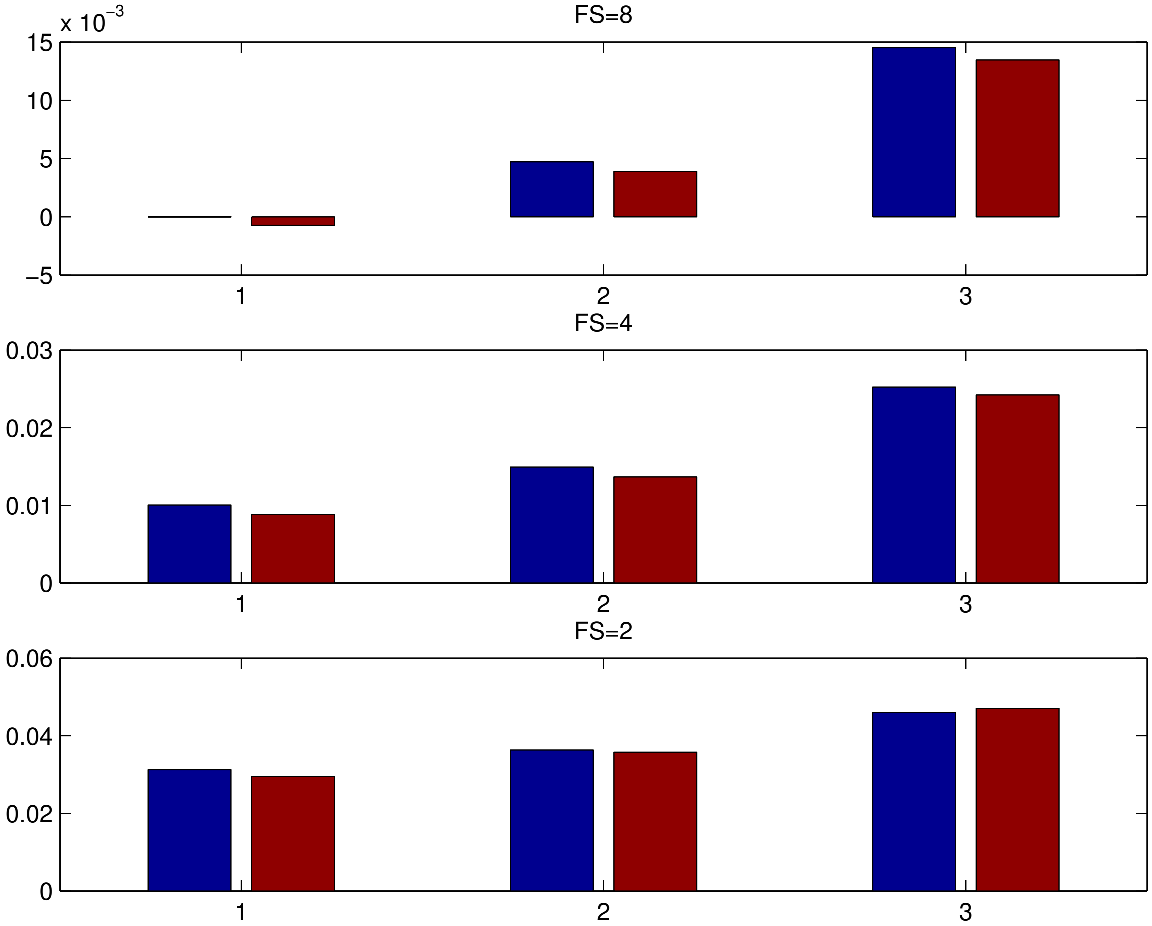
For MP3D, it is important to note that our hypothesis is supported in all cases. Also, from the common–flit–delay bars (red), we can see decreasing the flit size has a larger negative effect than increasing the flit delay. In other words, it's easer to make the program speed up with a wider network than by decreasing the flit delay.

3.3 LU

3.3.1 Description

LU factors a dense matrix into the product of a lower triangular and an upper triangular matrix. The data is distributed to exploit temporal locality and to reduce contention for blocks. [6]

3.3.2 Normalized Runtime



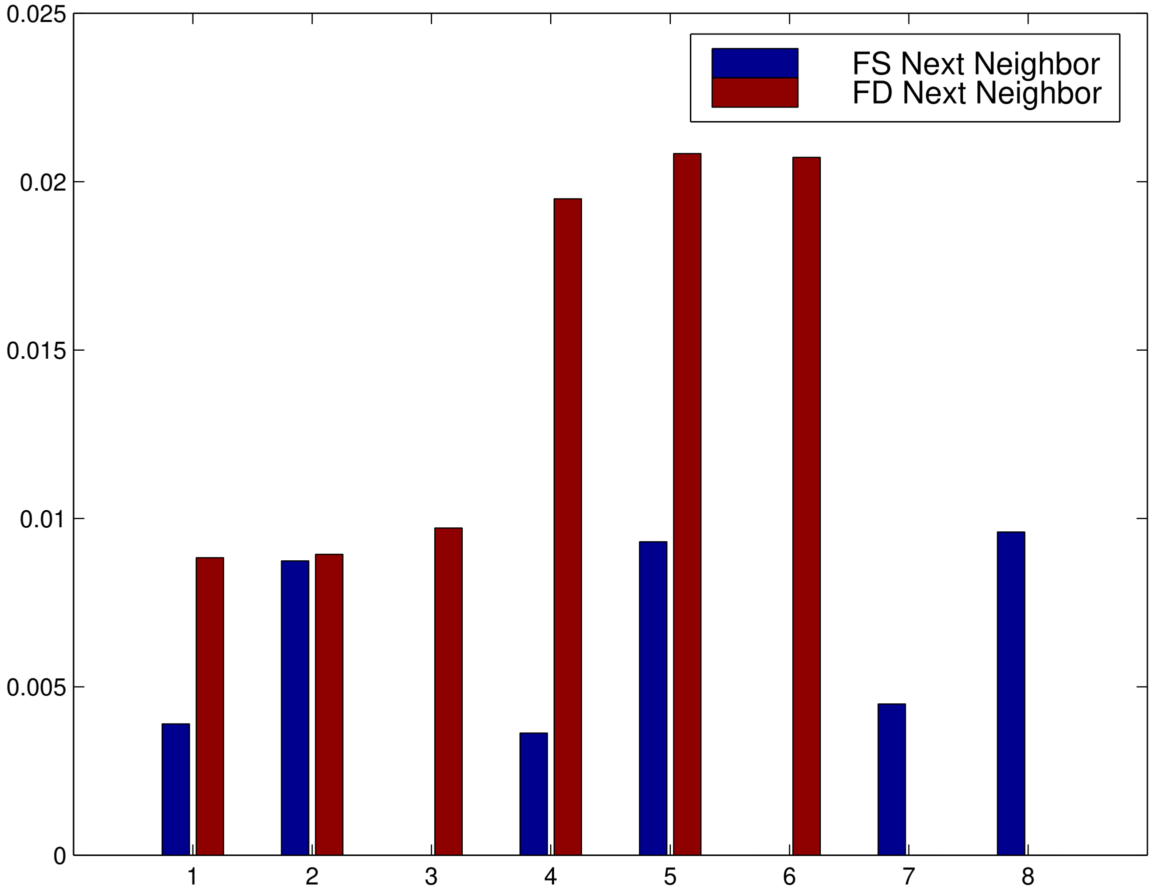
3.3.3 Runtime Discussion

Similar to FFT and MP3D, the protocol choice has little effect on the runtime when compared to flit delay or flit size. Flit size again has the largest impact, that is, all the runs with FS2 are much slower than those with FS4, and similarly for FS4 over FS2.

In LU, however, the MESI runs are not always faster than the MSI runs. In fact, the "baseline" run was not the fastest run in this set — that's why the first graph not based at 0! The reasons for this were described in the MP3D section.

The scale of runtime slowdown is much greater than that seen for FFT. The slowest LU run is 6% slower than its fastest run.

3.3.4 Neighbor Dominance



3.3.5 Dominance Discussion

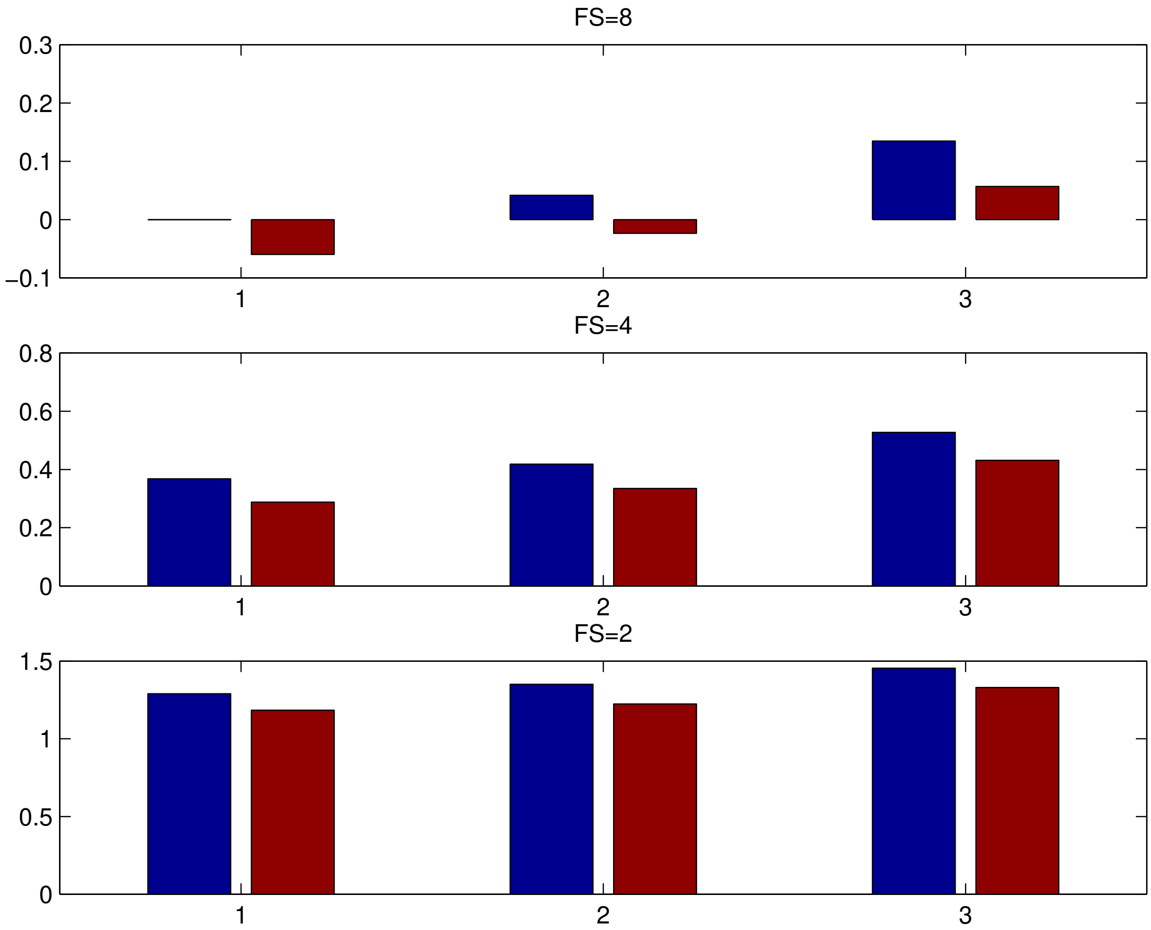
For LU, it is important to note that our hypothesis is supported in all cases. Also, from the common–flit–delay bars (red), we can see decreasing the flit size has a larger negative effect than increasing the flit delay. In other words, it's easer to make the program speed up with a wider network than by decreasing the flit delay.

3.4 RADIX

3.4.1 Description

RADIX is an integer sort kernel described by Blelloch *et al.* Each iteration requires all–to–all communication to send the local histograms to generate a global histogram, and then each processor uses this global histogram to permute its keys [6]. RADIX is data and communication intensive.

3.4.2 Normalized Runtime



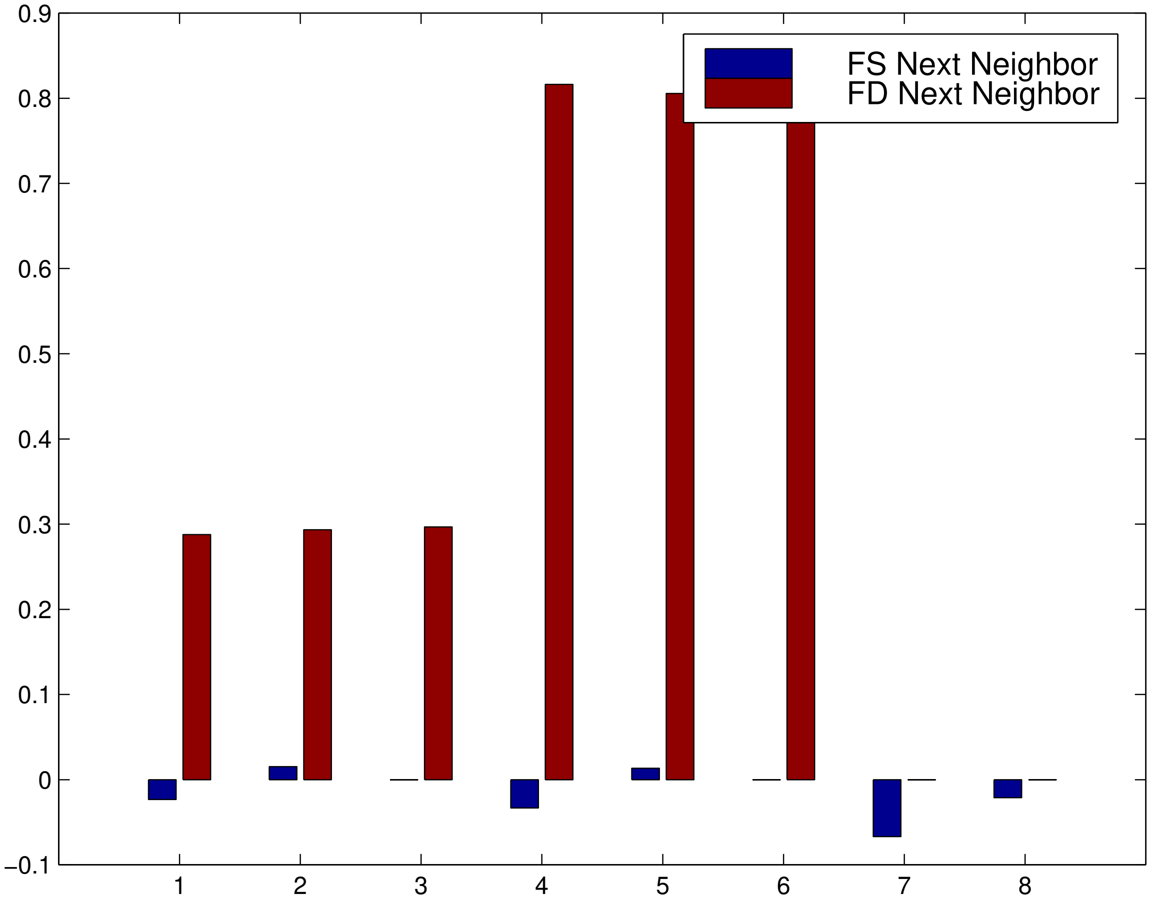
3.4.3 Runtime Discussion

Contrary to what we saw for the other benchmarks, the protocol choice has a large impact on RADIX. It is still true that improving flit size has the largest performance impact, but now protocol choice becomes as nearly as important as flit delay.

The MSI runs are always faster than the corresponding MESI run. This is because of two factors. The data-set is large, leading to much cache replacement, which creates more network traffic for those blocks in the E state. Secondly, when the global histogram is created, all processors must read it. The first time a value is read, it will be in the E state; when the next processor reads that value, the first processor must move to the S state, which requires a message.

The scale of runtime slowdown is over an order of magnitude more than that seen for FFT, MP3D, and LU. The slowest run is over 150% slower than the fastest run. We believe this is because of factors already mentioned: large data size, lots of communication, and communication on critical path.

3.4.4 Neighbor Dominance



3.4.5 Dominance Discussion

RADIX is the only benchmark which refutes our hypothesis. The negative values in the neighbor dominance graph show cases in which a lower–quality network running MSI has a faster runtime than a higher–quality network running MESI.

It is important to note, however, that this is only true in four cases (1, 4, 7, 8) and that in those cases the difference is between common–flit–size neighbors. In other words, even though flit delay does not dominate the effect of protocol in those cases, the effect of flit size *does* dominate the protocol choice.

Chapter 4

Conclusions

4.1 Hypothesis

We found that for the majority of the cases examined, network quality *does* dominate coherence protocol choice.

It was more strongly true for benchmarks which are not communication–intensive. The only benchmark which failed the hypothesis was RADIX, which requires a large amount of many–to–one and one–to–many communication every iteration.

The effect of protocol choice can be overcome by network quality. We found that increasing network width always dominated the protocol differences, while decreasing the network delay dominated protocol differences only "most of the time."

Further, it could be said that designers *should* dominate the protocol differences by the network quality because all programs will benefit from a better network, and not all benchmarks do better with MESI. As we found, some do much worse.

4.2 Protocol Importance

The choice of coherence protocol can be an important factor in system design. We saw that for the majority of the benchmarks we examined, protocol choice had little effect on performance, while network quality had the primary effect. However, for the one benchmark which had the worst slow–down, the protocol also had major impact on performance. Thus protocol choice must be paid special attention to during design: which protocol will perform better for the expected workloads? There is room here for future work.

4.3 Network Quality Importance

Network quality has the largest impact on performance. A faster network will improve all programs which communicate, with no performance cost — the cost involved is engineering and materials cost.

4.4 Application Dependence

Different applications have difference communication requirements. Some, like RADIX, are very network–intensive, while others tolerate low–quality networks with little performance degradation (FFT).

4.5 Future Work

We were surprised to see such a wide variance in maximum runtime difference between the benchmarks. This suggests a larger set of programs should be studied to get a better idea how a typical workload behaves.

This study has given us some idea about the relative importance of network quality as compared to protocol choice. The next step would be to compare the hardware and engineering costs to determine how trade-offs should be made.

Chapter 5

References

[0] *This Document.*
`http://www.cae.wisc.edu/~smithz/texts/ece757-report/report.ps`

[1] P. SWEAZEY and A. J. SMITH, "A Class of Compatible Cache Consistency Protocols and their Support by the IEEE Futurebus", *Proc. Thirteenth International Symposium on Computer Architecture*, Tokyo, Japan (June 1986), 414-423.

[2] S. V. ADVE and K. GHARACHORLOO, "Shared Memory Consistency Models: A Tutorial", *IEEE Computer*, 29, 12 (December 1996), 66-76.

[3] V. S. PAI, P. RANGANATHAN, and S. V. ADVE, "RSIM Reference Manual — Version 1.0", Rice University, Dept of ECE, Technical Report 9705, August 1997.

[4] Condor HTC Homepage. `http://www.cs.wisc.edu/condor/`, April 1998.

[5] J. P. SINGH, W. WEBER, and A. GUPTA, "SPLASH: Stanford Parallel Applications for Shared–Memory", Stanford University, Computer Systems Laboratory.

[6] S. C. WOO, M. OHARA, E. TORRIE, J. P. SINGH, and A. GUPTA, "The SPLASH-2 Programs: Characterization and Methodological Considerations", ISCA 1995.

[7] PAI, RANGANATHAN, ADVE, and HARTON, "An Evaluation of Memory Consistency Models for Shared Memory Systems with ILP Processors", ASPLOS 1996.