A Simple Capacity Model of Massively Parallel Transaction Systems

Neil J. Gunther

Systems Technology Laboratory
Pyramid Technology Corporation
San Jose, California, 95134-1702
nguntner@pyramid.com

This paper examines the scalability of "shared-nothing" architectures as the future platforms of choice for large, scalable, transaction processing systems. Capacity planners need some means of assessing throughput scaleup for these new platforms. We present a simple, semi-empirical, capacity model which expresses the possible scaleup for both shared-memory multiprocessors and massively parallel transaction systems based. A new concept, *super-seriality*, is introduced to to account for performance penalties due to the degree of shared data in commercial database workloads. Super-seriality has the effect of inducing a premature *maximum* in the scaling curve beyond which it is counter-productive to add further processors. This results in unexpected limitations in the scalability of "shared-nothing" architectures executing transaction processing workloads. Scaleup predictions are presented for some existing transaction processing platforms.

1. INTRODUCTION

Massively parallel processors (MPP's) are emerging as commercially viable candidates for displacing conventional mainframe computer systems in the realm of large-scale data processing and transaction processing tasks.

We present an analysis of transaction processing scalability on MPP architectures with the objective of providing the future capacity planner with a rational means of assessing plausible sizing limitations during the expected onslaught of promotional statements about thousands of commodity processors offering thousands of aggregate MIPS (Mega Instructions Per Second) in computational horsepower.

To aid the reader, unfamiliar with the terminology that typically accompanies this emerging technology, a brief GLOSSARY section is included at the end of this paper.

David DeWitt and Jim Gray [DeWitt&Gray92] have suggested that this re-emergence is coupled with the widespread commercial success of relational database management systems (RDBMS's). Relational queries are eminently suited to parallel execution on multiple data streams. Many MPP architectures can be classified as "shared-nothing" [Stonebraker86] in that each of the potentially hundreds or even thousands of processing nodes possesses their own local memory and disk storage devices. They therefore can operate quite independently except for the occasional exchange of data across a high-performance interconnection network.

MPP architectures were pioneered largely in the scientific and research communities where the focus is on improving the speedup of scientific codes via parallelizing compilers. Commercial database applications were at first thought to have only limited and specialized potential but are now re-emerging as RDBMS vendors turn their attention toward utilizing the wealth of commodity MIPS available in MPP's.

Shared-nothing architectures[1] get their name by virtue of minimizing the sharing of hardware resources leading to the claim that these platforms can be scaled up to hundreds and potentially thousands of processors (e.g., CM5, KSR1, nCUBE, Teradata) This massiveness in computational horsepower, in turn, leads to claims and expectations of near-linear scalability on transaction processing workloads.

In a marketplace where such claims are likely to become more vociferous, we attempt to sound a cautionary note: near-linear scalability on multiprocessor systems should not be heeded without reference to the amount of shared data and code present in the workload. Shared data and code abound in the currently available commercial

1. For a readable comparison of some commercial implementations see [Gottlieb92].

RDBMS products. This sharing is driven by the desire for:

- A general purpose programming model in high-level database applications.
- Ease of production release and maintenance through the use of shared data structures.

Since the degree of sharing in RDBMS architectures is currently non-zero and is unlikely to improve in the near future, we must live with the constraint that linear scaling for transaction processing systems is as practically unattainable as it has been for massively parallel scientific workloads.

Broadly speaking, database transaction workloads exhibit a higher degree of serialization than is the case for fine-grain parallelism. As we have discussed elsewhere [Gunther92], Symmetric Multi-Processor (SMP) systems running workloads with a high degree of shared data can exhibit measureable "roll-off" [Thakkar90], rather than just saturation in performance. The question for the capacity planner then becomes: "What degree of sublinearity is acceptable for a given transaction system?"

We present a capacity saleup model that accounts for the increased amount of serialization due to the existence of shared data. As we shall see, the proposed model exhibits a maximum in the scaling curve; the onset of which can occur long before the expected system saturation level! Moreover, since the models we present here are nowhere dependent on any assumptions about bus or interconnect technologies, they already draw attention to software inefficiencies.

2. SCALING MODELS

We study processor scalability from the point of view of the execution of a workload that is scaled to the number of available processors. [DeWitt&Gray92] refer to this as "scaleup" rather than "speedup" (See GLOSSARY section).

2.1 PEFORMANCE METRICS

Transaction scaleup is ideal for MPP systems since each transaction is a relatively small, independent, task that can run on separate processors. It is for this reason that the TPC council has chosen a special set of scaling rules for database sizing as more processing power is applied to the workload. The capacity metric used in this paper is the throughput rate (R) measured in Transactions Per Second (TPS) and the workload is equivalent to that found in the TPC Benchmark B™ [Gray91].

We compare the TPS rate achieved by N processors (R_N) relative to that achieved by a single processor (R_1). The transaction throughput scaleup is then given by the ratio: $\frac{R_N}{R_1}$.

We assume that each processor is optimized for process concurrency i.e., enough transaction generation processes are present to utilize the available cycles on a processor and yet not cause a higher than necessary context-switch rate. Only one transaction generation process runs at a time on a single CPU. In general, the processor will not be doing any useful work if it must:

1. wait for a bus transfer from another processor or memory.
2. wait for I/O completion
3. serialize on an RDBMS lock or latch.

2.2 DUAL-PROCESSOR SCALEUP

Suppose a uniprocessor completes t_1 transactions in an elapse time, τ. Its throughput is then given by:

$$R_1 = \frac{t_1}{\tau}.$$

Doubling the workload, we would expect the uniprocessor to complete $2t_1$ transactions in twice the time, i.e., 2τ.

Similarly, we would expect a dual processor (N = 2) system to complete $t_2 = 2t_1$ transactions in the same time it takes the uniprocessor to complete t_1 transactions. Measurements on shared-memory multiprocessor systems show, however, that t_2 is generally less than $2t_1$ in time τ. In other words, the dual processor takes *longer* to complete $2t_1$ transactions! The two processors are said to "interfere" with one another's ability to execute independently.

Let us denote the additional elapse time as a fraction σ of the uniprocessor time, i.e., $\sigma\,\tau$, where $0 \le \sigma \le 1$. Then, the dual processor throughput is given by:

$$R_2 = \frac{2t_1}{\tau + \sigma\,\tau}.$$

Dual-processor scaleup can now be expressed as:

$$R_2 = \left[\frac{2}{1+\sigma}\right] R_1,$$

where we have cancelled the common factors of τ in the quotient. What is the signifiance of this equation?

Clearly, for any non-zero value of σ, the throughput capacity of the dual-processor system will be *less* than twice that of uniprocessor system. Suppose, for example, that $\sigma = 0.03$ then the scalability formula tells us that the effective throughput of the dual-processor will only be 1.94 times that of the uniprocessor. Therefore, if the uniprocessor is capable of 100 TPS, the dual-processor will achieve only 194 TPS - not 200 TPS. This will be the case if the two processors spend 3% of the elapse time interfering with each other's ability to generate transactions. The parameter, σ, is a measure of this interference. But what is the nature of this interference?

In SMP transaction processing systems there are many points of serial contention in both the hardware (e.g., the memory bus), and the DBMS (e.g., critical code sections) where one processor must wait for the other to complete before it can continue to execute transactions. This time spent contending for shared resources is the dominant reason for dilating the dual-processor execution time. The impact of this effect may not be cause for concern in a dual-processor system but if the trend holds as more processors are added to the system then cumulative effects at the high-end (large-N) may have significant consequences. Keep in mind that MPP hardware is often technically capable of supporting many hundreds and even thousands of processors. To determine the potential impact, for large-N systems, we need to generalize the dual-processor capacity equation.

2.3 GENERALIZED SCALEUP

Another way of looking at dual-processor scaleup is that it tells us about the effective number of processors doing real work i.e., actually executing transactions. Let us therefore define a generalized capacity function, $C(N)$ by analogy with our discussion in section 2.2.

$$R_N = C(N)\, R_1.$$

At this point in our discussion, we do not know the form of $C(N)$ for an arbitrary number of processors. What we can expect, however, is that the following assumptions hold:

1. $C(1) = 1$, trivially.
2. The fraction of time, σ, will remain constant but scale as the number of processors increases according to $\sigma(N) = \sigma f(N)$ where $f(N)$ is to be determined.
3. Any general capacity function should contain dual-processor scaling, $C(2) = \frac{2}{1+\sigma}$, as a special case.

With these constraints in mind, we write a general capacity function for an arbitrary number of physical processors (N) as:

$$C(N) = \frac{N}{1 + \sigma f(N)}$$

where $f(N)$ is an undetermined function of the number of physical processors in the system. It should be clear by now that no matter what choice is made for $f(N)$, the scaleup will be nonlinear and in general it will be sublinear for non-zero values of σ. Furthermore, it is reasonable to expect that the contributions to the time dilation are piece-wise additive. We therefore assume, for simplicity, that $f(N)$ is polynomial in N.

Elsewhere [Gunther92], we have considered two important forms for $f(N)$ in an attempt to model SMP transaction processing performance. We summarize those results here.

2.4 OPTIMISTIC CAPACITY MODEL

In this model, $f(N) = (N - 1)$, is linear in N. Hereafter, we shall refer to this form as the Optimistic model[2]. The explicit form of $C(N)$ for this model is:

$$C(N) = \frac{N}{1 + \sigma(N - 1)}$$

This Optimistic model produces scaleup in which $C(N)$ is monotonically increasing for all values of N and asymptotically approaches saturation at σ^{-1}. As we will demonstrate in the next section, this model is too optimistic in its prediction of multiprocessor throughput capacity. There are other effects contributing to the execution time-dilation which are not accounted for in the Optimistic model.

2.5 SUPER-SERIAL MODEL

In this model, $f(N) = (N - 1) + \lambda N(N - 1)$, is quadratic in N and $0 \le \lambda \le 1$ is another independent parameter. Hereafter, we shall refer to this form as the Super-Serial model. The explicit form of $C(N)$ is:

$$C(N) = \frac{N}{1 + \sigma(N - 1) + \sigma\lambda N(N - 1)}$$

This model has been found to provide a broader match with measured data, in part, because it approaches zero

2. The functional form of $C(N)$ is identical to that found in Amdahl's law [Amdahl67] for parallel speedup of a fixed, single task. The reader is cautioned against taking the analogy too far since our models are for multi-user scaleup.

asymptotically as N^{-1} (Fig. 1). Moreover, the Super-Serial model contains the Optimistic model in the special case of $\lambda = 0$. Both of these polynomial forms are expressed in terms of $(N-1)$ so as to satisfy the first of the assumptions above i.e., $C(1) = 1$ when $f(1) = 1$.

Dividing $C(N)$ by N, we can re-express the denominator as:

$$\frac{1}{N} + \sigma\left[\frac{N-1}{N}\right] + \sigma\lambda(N-1)$$

This form reveals a convincing physical interpretation for the various contributions to C(N). We may think of each term as representing, respectively, one of the *Three C's*: concurrency, contention and coherency. The first term represents the degree of concurrency available in the code. The second term represents contention for shared resources. The third term is more obscure but we have argued in [Gunther92] that it can be identified with additional latency due to some fraction, λ, of processor activity invalidating shared, writeable, data.

We call this latter effect *super seriality* to emphasize its connection with the existence of serial contention (seriality) and the extension of its effects. Note that this statement is captured by virtue of the super-serial term vanishing when $\sigma = 0$.

Figure 1 depicts the general characteristic of $C(N)$ for N ranging from 1 to 100. σ is held constant while λ ranges from zero (Optimistic Model) to a nominal 0.1. Note the gradual appearance of a broad maximum in the curve for λ=0.01 and furthermore, how it both narrows and moves toward the origin as λ increases. The most significant difference between the two models is the presence of a maximum in the Super-Serial scaling curve at the critical value:

$$N_c = \sqrt{(1-\lambda)/\lambda\,\sigma}$$

For a fixed value of σ, the position of the maximum is determined by the value of λ according to:

1. $N_c \to \infty$ as $\lambda \to 0$; (Optimistic model limit)
2. $N_c \to 0$ as $\lambda \to 1$

Compare this with the simpler asymptotic behavior of the Optimistic model in Fig. 1.

In the remainder of the paper, we validate the Super-Serial model against measured SMP data and extrapolate the results to MPP performance.

3. NUMERICAL RESULTS

First, we turn to the validation and application of our general capacity function in the context of transaction processing workloads. The only comprehensive data, available to us, is for the TPC Benchmark B™ workloads [Gray91]. Although TPC-B is not typical of all transaction processing workloads, it does provide a common baseline by which to make certain scaling comparisons. It is sufficient for our discussion of sharing latencies because there are common functional parts of the RDBMS that must be exercised by all the benchmark transactions.

Table 1.

Shared-Memory Multiprocessor Scalability

Parameters: $\sigma = 0.029$, $\lambda = 0.05$

	Measured TPS		Predicted TPS			
CPUs	Sample TPS	Percent Sublinear	Super Serial	Error	Optimistic Model	Error
1	–		77.8		77.8	
2	150.0	3.7	150.9	0.6	151.3	0.9
3	–		218.9		220.7	
4	282.0	9.4	281.9	0	286.4	1.6
5	–		339.8		348.7	
6	395.0	15.4	392.8	-0.6	407.8	3.2
7	–		441.0		464.0	
8	496.0	20.4	484.7	-2.3	517.4	4.3
9	–		523.9		568.4	
10	–		559.0		617.0	
11	–		590.3		663.4	
12	627.0	32.9	617.9	-1.5	707.8	12.9
13	–		642.2		750.2	
14	671.0	38.4	663.5	-1.1	790.9	17.9
15	–		681.9		829.9	
16	–		697.7		867.3	
17	–		711.2		903.2	
18	704.0	49.8	722.6	2.6	937.7	33.2
19	–		732.0		970.9	
20	–		739.6		1002.9	
21	–		745.7		1033.6	
22	–		750.4		1063.3	
23	–		753.8		1091.9	
24	–		756.0		1119.6	
25	N/A		757.2		1146.2	
26	"		**757.5**		**1172.0**	
27	"		757.0		1196.9	
28	"		755.7		1221.0	
29	"		753.8		1244.4	
30	"		751.4		1267.0	

To analyze the data[3], we developed a fitting program

3. The measured data samples in Tables 1 and 2 come from a variety of sources and therefore do not provide the complete set that otherwise would be desirable for our modelling discussion. Negative error values indicate under estimation by the model.

called, qcomp, to assist in determining appropriate values of σ and λ from sampled, low-end throughput data. The solid curves in Figures 2 and 3 were generated in this way. The program uses a standard linear least-squares fit [Press88] to the power series expansion:

$$C(N) = N + \sigma N(N-1) - \sigma^2 N(N-1)^2 + \lambda\sigma(N-1)N^2 - O(N^4)$$

It turns out that near N_c, the maximum in $C(N)$ can be predicted quite accurately from just the first two terms. This is a reasonable approximation because for $N \leq N_c$ the scaling curve is well-approximated by an inverse parabolic function, while for $N > N_c$ the curve is dominated by the coherency term carried with the factor λ.

Table 2.

Massively Parallel Scalability

Parameters: $\sigma = 0.028$, $\lambda = 0.023$

	Measured TPS		Predicted TPS			
CPUs	Sample TPS	Percent Sublinear	Super Serial	Error	Optimistic Model	Error
1	24.3	0.0	24.3	0	24.3	0
2	44.3	8.8	48.5	9.5	48.5	9.5
3	–		72.5		72.5	
4	80.5	17.2	96.3	19.6	96.4	19.8
5	–		120.0		120.2	
6	–		143.5		143.8	
7	–		166.9		167.3	
8	157.8	18.8	190.0	20.4	190.7	20.8
9	–		213.0		214.0	
10	–		235.8		237.1	
–	...					
64	1073.0	31.0	1086.1	1.2	1323.9	23.4
...	...					
120	N/A		1301.3		2192.4	
121	"		1301.6		2206.1	
122	"		1301.9		2219.7	
123	"		1302.0		2233.3	
124	"		1302.2		2246.8	
125	**"**		**1302.2**		**2260.2**	
126	"		1302.2		2273.6	
127	"		1302.1		2286.9	
128	"		1301.9		2300.2	
129	"		1301.6		2313.4	
130	"		1301.3		2326.6	

3.1 MULTIPROCESSOR SYSTEM

Table 1 compares measured and predicted transaction throughput, R_N, for each of the models discussed. The column labelled *Sample TPS* shows measured performance data taken from a Pyramid MIServer™ running ORACLE 7™ with a workload equivalent to the TPC-B™ benchmark. These data are accurate to about ± 3%. The next column is the degree of sublinearity (relative to a single CPU) expressed as a percentage.

Our approach assumes that single or dual processor measurements are generally more accurate than those taken on systems configured with a large number of CPUs. In the current MIServer™ ES-Series product line, a maximum of 24 R3000A processors are available. Up to 6 of these processors are usually dedicated to handling OLTP network traffic and RDBMS logging processes. Therefore, only 18 processors were available for executing database transactions. The Super-Serial model predicts a capacity roll-off around $N_c = 26$; just beyond the number of CPU's that can be physically slotted into the ES backplane. Altough this might be viewed as a Marketing win, it is unfortunate for the purposes of validating $C(N)$ beyond the maximum.

Due to this limitation, we have validated our capacity models against alternative data available for the Sequent Symmetry™ which has been measured for different cache protocols namely, "write-through" and "copy-back" [Thakkar90]. Both protocols refer to main memory transfers across the shared memory bus. The former protocol updates main memory (across the memory bus) on every update while the latter only does so on a read. In particular, our capacity function $C(N)$ accounts accurately for the maximum in the write-through data with $\sigma = 0.03$ and $\lambda = 0.15$. The copy-back data is fitted to $\sigma = 0.01$ and $\lambda = 0.02$. The data in Figure 2 show an excellent fit to the Super-Serial model. Furthermore, since these data do *not* belong to a database workload, they lend support to the idea that super-seriality occurs in the presence of any shared, writeable data and therefore exists as a quite general phenomenon.

We have just indicated that hardware attributes such as cache protocols and bus bandwidth can play a significant role on determining system scalability. In general, however, for a hardware platform with fixed protocols, and bus characteristics the values of σ and λ will thereafter be determined by software attributes as reflected in: resident set size, cache footprint, lock management policies, contention for internal buffers and length of critical sections.

3.2 MASSIVELY PARALLEL SYSTEM

Table 2 presents data for the nCUBE2™ running ORACLE 6.2 with the TPC-B workload. The low-end data are informal and supplied by Oracle Corporation and probably responsible for error magnitudes that are higher

than for the SMP case. The implication is that these were inferior TPS values that could have been improved on with additional tuning effort. The 64-node measurement is a bona fide TPC disclosed result [Oracle91]. The projections beyond 64 nodes were obtained using `qcomp` to calculate $C(N)$.

Figure 3 shows the SMP and MPP data (from Tables 1 and 2) together with capacity projections given by $C(N)$. The reader is encouraged to cover up the right-hand side of the figure and note how the initial perception of "linearity" must be mentally readjusted as the cover is removed and the roll-off appears around $N_c = 125$ processors for the nCUBE2. The reader should observe in Figure 3 that although σ is approximately the same for SMP and MPP, λ for the MPP is about half the SMP value which gives rise to a broader scaling curve and higher N_c.

We note, in passing, that the aggregate MIPS in the MPP case is around 500 while the more recent SMP platform has around 650 aggregate MIPS available on one third as many processors. This is a consequence of the creeping commodity MIPS envelope between two architectures that have only a year between their respective release dates. It is also noteworthy that the nCUBE result is still the only TPC measurement on an MPP architecture after more than two years; yet another signal of the complexities involved in running database workloads on MPP's.

4. CONCLUSIONS

We have presented a simple analytic model for capacity scaleup on SMP and MPP multiprocessor transaction systems. The notion of super-seriality seems to provide a simplified account of performance roll-off and has been demonstrated to have some reliable predictive power. Clearly, such naive assumptions as we have used could be further refined to produce a more sophisticated model (see Appendix) but this was not our purpose here.

Performance roll-off is a symptom of super-seriality. This effect can arise, for example, when a processor is primarily serialized spinning on a lock to access shared data and is further serialized after obtaining the lock by the need to fetch cache lines (across the memory bus) that have been invalidated by the preceding action of other processors on that data. In SMP's, super-seriality is a latency effect due to the presence of a memory hierarchy e.g., local caches and global shared memory. It is similar in character, but different in time-scale, to the performance roll-off caused by virtual-memory paging ("thrashing") when the number of user-processes competing for finite main memory increases beyond the multi-programming level. At that point, additional latency is incurred because the current pages in memory must be replaced by pages that are in auxiliary memory - typically a slower disk. This effect has been modelled as a load-dependent central-server queueing network which has computable transient behaviour [Gunther89] and Appendix (this paper). In fact, due to the presence of a performance optimum in both cases, the throughput curves for the virtual memory model show a remarkable, qualitative, similarity to the super-serial scaling curves presented here.

To illustrate the notion of super-seriality, we drew upon an example which referred to shared data accesses. A similar effect is also present in instruction caches. Although instructions are read-only, serial contention for critical sections in the DBMS (e.g., manipulating an LRU list) may induce instruction fetching into the local cache - at a lower miss-rate than for shared data.

Our capacity model is a simple semi-empirical model that relies on low-end throughput measurements to determine the two parameters σ and λ. We have successfully developed a numerical fitting program to assist in this process. Even with these limitations, the current model:

1. Directs our attention to the quantitative significance of resource sharing in both SMP and MPP architectures.
2. Requires only two parameters which ultimately can be modelled explicitly (see Appendix).
3. Alerts capacity planners to the potential for an intrinsic maximum in the scaleup function due to super-seriality.
4. Places the onus on the RDBMS vendors to achieve the promise of shared-nothing architectues.

These are areas of further research.

5. APPENDIX: QUEUEING NETWORK MODEL

In this appendix, we outline a derivation of our semi-empirical model based in the theory of queueing network models. We employ the notatation found in [QSP84].

The importance of this more theoretical approach is that it demonstrates the reasonable degree of approximation contained the semi-empirical formula and it shows how it

is possible to determine the two parameters, σ and λ from fundamentally measurable quantities.

A more detailed discussion would fall well outside the scope of this paper and therefore will be presented elsewhere.

Commencing with the Response Time law [QSP84] for throughput X,

$$R(N) = \frac{N}{X} - Z,$$

we can write our capacity scaleup function (after some tedious algebraic manipulations) as:

$$C(N) = \frac{N}{1 + \frac{N-1}{N}\left[\frac{\sum Q_k D_k}{Z + D}\right]} \qquad (A)$$

where Q_k is the average queue-length at node k (e.g., the system bus or a memory module) and D_k is the service demand at that node. Z is the mean time spent executing at each of the N processor (delay) nodes and D is the total service demand across all queueing nodes. The summation in the denominator is over all k queueing nodes.

5.1 OPTIMISTIC MODEL

For a standard queueing network described by this equation, the only quantity that depends on N is Q_k. All other quantities are constant. Furthermore, one can show that only the "bottleneck" node (the one with the largest service demand) has a queue that grows with N. It turns out that this growth is always *linear* in N i.e., $Q_k = N$ in what we call the "heavy-traffic" (large-N) regime. Let us suppose for this discussion that the bottleneck device is the memory with a constant service demand, D_{mem}.

Substituting all this into the above equation, gives:

$$C(N) = \frac{N}{1 + (N-1)\left[\frac{D_{mem}}{Z + D_{mem}}\right]} \qquad (B)$$

Note that this equation is identical in form to the Optimistic model with

$$\sigma = \frac{D_{mem}}{Z + D_{mem}}.$$

System capacity will therefore saturate at

$$C = \frac{Z + D_{mem}}{D_{mem}}, \text{ as } N \to \infty.$$

Compare with the top curve in Fig 1.

5.2 SUPER-SERIAL MODEL

The corresponding derivation of the Super-Serial model hinges off the existence of a queueing node at which the service demand is no longer constant but rather depends on the system load - the number of active processors (N). In this way we incorporate the *super-seriality* effect presented in this paper.

If we call the super-serial service demand, D_{ss}, and assume for the moment that its growth is linear in N (but less dramatic than the queue growth at the bottleneck device in a standard queueing network, we will pick up a term going like $(N-1)N$ which is quadratic in the denominator of C(N). This is a consequence of the crossover in the growth of queue lengths from bottleneck queueing to queueing at the super-serial device.

6. REFERENCES

[Amdahl67] G. Amdahl, "Validity of the Single-processor Approach to Achieving Large-scale Computer Capabilities," *Proc. AFIPS Conf.*, 483-85, Apr 1967.

[DeWitt91] D.J. DeWitt, "The Wisconsin Benchmark: Past, Present, and Future," in [Gray91].

[DeWitt&Gray92] D.J. DeWitt and J. Gray, "Parallel Database Systems: The Future of High Performance Database Processing," Univ. Wisconsin-Madison, CS Technical Report #1079, 1992.

[Gottlieb92] "Architectures for Parallel Supercomputing," Transcript by Allen Gottlieb, NYU, New York, 1992. Available via `ftp: cs.nyu.edu:pub/tech-reports`.

[Gray91] *The Benchmark Handbook for Database and Transaction Processing Systems*, ed. J. Gray, Morgan Kaufmann, San Mateo, CA, 1991.

[Gunther89] N.J. Gunther, "Path Integral Methods for Computer Performance Analysis," Information Processing Letters, v32, #1, 7-13, 1989.

[Gunther92] N.J. Gunther, "Assessing Transaction Processing Scalability in Shared Memory Multiprocessors," Pyramid Performance Notebook, Nov 1992.

[Oracle91] *TPC Benchmark B Full Disclosure Report for the nCube 2 Scalar Supercomputer Model nCDB-1000 Using ORACLE V6.2*, Oracle Corp., 1991.

[Press88] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C*, Cambridge University Press, 1988.

[QSP84] *Quantitative Systems Performance: Computer System Analysis Using Queueing Network Models*, E.D. Lasowska, J. Zahorjan, G.S. Graham and K.C.Sevcik, Prentice-Hall, New Jersey, 1984.

[Stonebraker86] M. Stonebraker, "The Case for Shared Nothing," Database Engineering, vol. 9, no. 1, 1986.

[Thakkar90] S.S. Thakker, "Performance of Symmetry Multiprocessor Systems," in *Cache and Interconnect Architectures in Multiprocessors*, eds. M.Dubios and S.S. Thakker, Kluwer Academic, 1990.

7. GLOSSARY

For the reader's convenience we provide a brief glossary of terms and acronyms used throughout this paper.

- **C(N):** Scaleup capacity function.
- **CPU:** Central Processing Unit - increasingly, a commercial microprocessor.
- **λ:** Super-seriality parameter with range [0, 1]. Associated with second-order coherency latency. Responsible for the premature maximum in the scaleup function C(N).
- **Latch:** Short-duration lock used internally by the RDBMS.
- **Lock:** Long-duration synchronization mechanism used on shared data structures.
- **MIPS:** Mega (10^6)Instructions Per Second.
- **MPP:** Massively Parallel Processor.
- **N:** Number of processors (CPU) in an SMP or the number of processing nodes in an MPP.
- **Node:** MPP CPU.
- **OLTP:** On-Line Transaction Processing.
- **RDBMS:** Relational Database Management System.
- **Scaleup:** Ability of a greater number of processing nodes to accommodate a proportionaly greater workload in a fixed amount of time.
- **Shared-nothing:** Architecture which minimizes sharing of hardware resources. Typical of MPP's.
- **σ:** Seriality parameter with range [0, 1]. Associated with serialized contention latency.
- **SMP:** Single-bus shared-memory Multi-Processor.
- **Speedup:** Reduction in elapsed time obtained by executing a fixed-size workload on a greater number of processing nodes.
- **TPC:** Transaction Provcessing Performance Council.

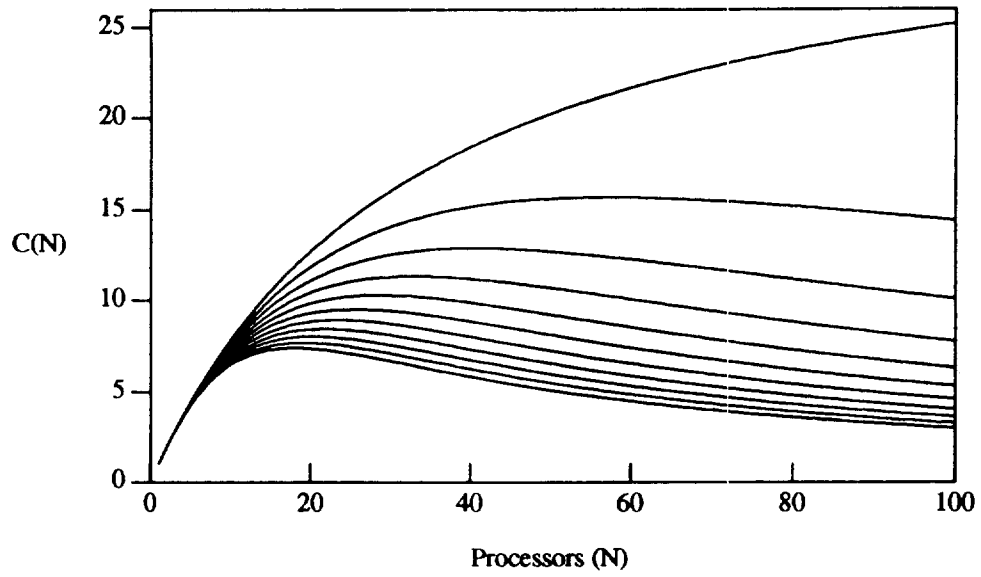

Figure 1. Characteristic C(N) curves for the super-serial model with σ held constant and λ ranging from 0.00 (top curve), to 0.10 (bottom curve).

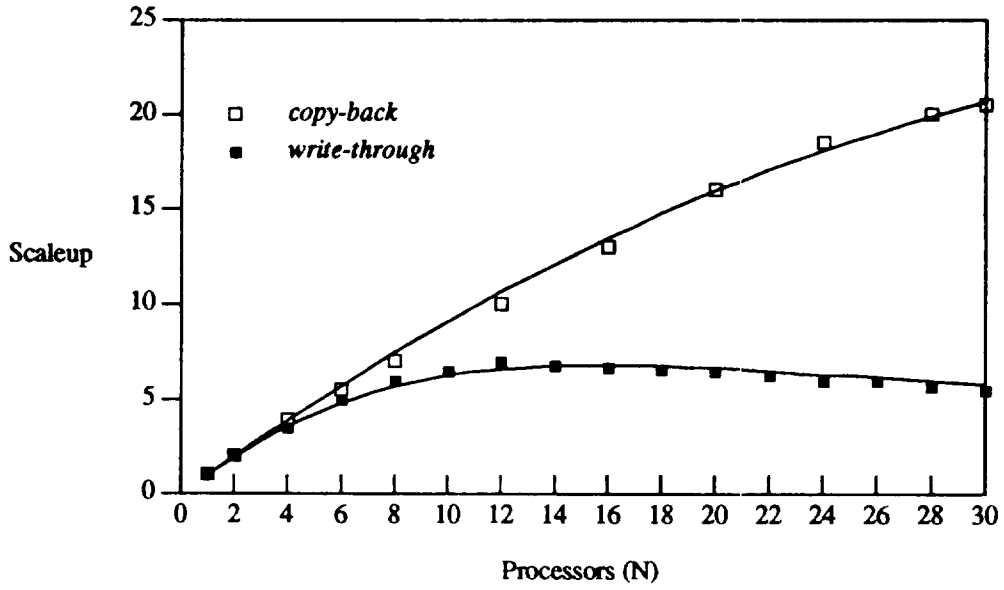

Figure 2. Architectural impacts on scaleup. Measured scaleup (squares) for different cache policies for a non-OLTP application with a significant degree of shared data. Solid curves correspond to Super-Serial C(N) for these data.

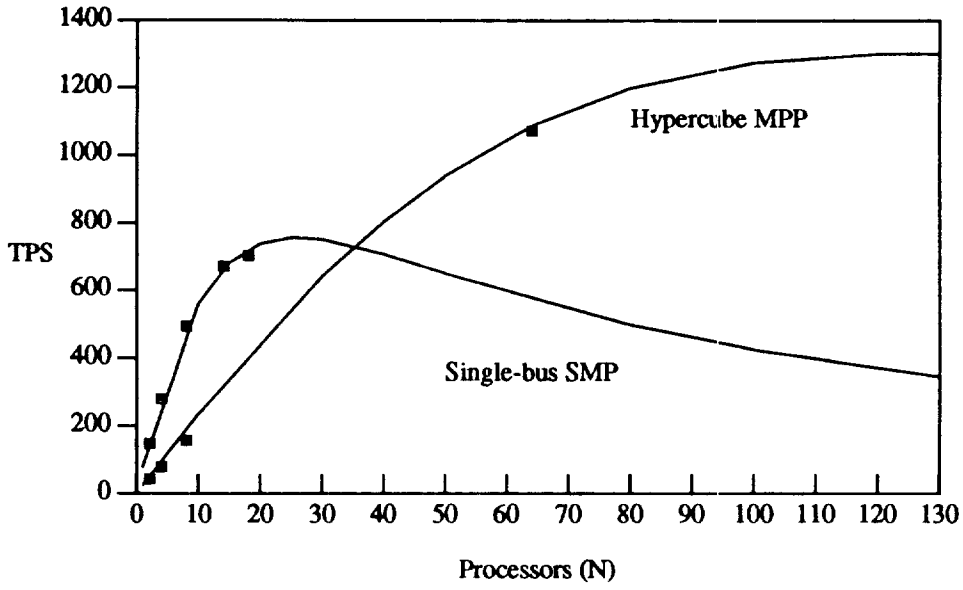

Figure 3. Comparison of predicted throughput capacity based on C(N) for the super-serial model (solid curves) with measured data (squares) for both SMP (from Table 1) and MPP (from Table 2) architectures.