

Architecture with No Economic Efficiency Barrier of Scaling

Scaling with Zero Marginal Costs

Transactional Cascade® Technology Paper

Ivan Klianev, Managing Director & CTO



January 2010

Copyright © 2010 Transactum
All rights reserved

Abstract

Scaling of business processes is not linear. The main reason is the insufficient parallelism of database transactions. Non-linearity appears in the diminishing marginal processing capacity and *Marginal Costs* barrier, and explains the increasing process latency and *Marginal Latency* barrier. Applications scale well where database operations are dominated by data reading and could be partitioned over multiple physical databases. Good examples are social networking applications. However, business processes with multiple aggregations are so far unsuccessful with scaling for Internet. The lack of success is attributed to the *Marginal Costs* barrier and its appearance with these processes at the earlier stages of scaling. Transactional Cascade® workflow is the only known technology that has zero marginal scaling costs and eliminates the *Marginal Costs* barrier. In addition to saving large infrastructure budgets and energy costs, Transactional Cascade® provides more room for scaling as it is limited only by the *Marginal Latency* barrier, which appears at a considerably later stage

Table of Contents

Abstract	2
Table of Contents	3
Executive Summary	4
1. Barriers of Scaling	5
1.1. Non-Linearity of Scaling	5
1.2. Reason for Non-Linearity	6
1.3. Marginal Costs Barrier	6
1.4. Marginal Latency Barrier	7
2. Non-Linearity Curve	8
2.1. Function of Data Processing Nature	8
2.2. Effect of Increased Parallelism	9
3. Barriers in Different Application Categories	10
3.1. Marginal Costs Barriers	10
3.2. Marginal Latency Barriers	11
4. Middleware Architecture and the Barriers	12
4.1. Workflow-Enabled Concurrency of Database Transactions	12
4.2. CPU Utilization and Marginal Costs of Scaling	14
4.3. Scaling With No Barriers	15
4.4. Scaling With No Marginal Costs Barrier	15
4.5. Scaling With A Marginal Latency Barrier	16
5. Conclusion	18

Executive Summary

The different nature of data operations in different categories of business processes defines the appearance of scaling barriers at different stages of scaling. This paper explains the reasons behind successful Internet scaling of social networking and the lack of success with enterprise applications. It shows how Transactional Cascade® technology could help with Internet scaling of any business process using limited hardware, software licenses, and energy consumption.

Scaling of a business process application is not linear. The scaling curve represents the increase of application processing capacity that results from adding a new process instance. Number of process instances reflects the number of user interactions an application can process concurrently. The difference between hypothetical linear and the realistic non-linear increase of processing capacity represents the increase of process latency with scaling.

Marginal processing capacity diminishes with application scaling, while marginal process latency increases. These explain the existence of two barriers of scaling. *Marginal Costs* barrier stands for maximum acceptable costs for additional infrastructure per new unit of processing capacity. Marginal Latency barrier refers to maximum acceptable increase of the average users-experienced latency per added infrastructure for a new concurrent user.

When using workflow-enabled middleware capable only of CPU-based parallelism, the first barrier that appears with scaling is the *Marginal Costs* one. With different application categories, it appears at different stages of scaling, as it is a function of data processing nature. Explanation of successful Internet scaling of social network applications is in their domination by database read operations and the ability to partition over multiple physical databases. Both facts enable large scaling without hitting the *Marginal Costs* barrier.

Processes with mainly database read operations have steeper scaling curve because of the higher concurrency per database and higher parallelism with multiple physical databases. In contrast, the unsuccessful so far Internet scaling of business process applications is attributed to the existence of multiple aggregations that require updating with serialized transactions and inability for database partitioning. Both facts restrict scaling as the *Marginal Costs* barrier appears at very early stages.

Transactional Cascade® workflow technology creates high volume of new business process instances with the same hardware and the same software licenses. Having zero marginal costs, scaling with it does not experience Marginal Costs barrier and might continue up to the absolute maximum of achievable processing capacity, but there is no practical value of doing it. Scaling creates value when it is performed up to the *Marginal Latency* barrier.

Scaling with Transactional Cascade® workflow technology saves huge infrastructure budgets and energy costs, and enables scaling up to the point of maximum acceptable process latency. It eliminates the existence of economic efficiency barriers of scaling and enables a new breed of conceptually feasible Internet-scale business processes to become economically feasible application projects.

1. Barriers of Scaling

Scaling an application means expanding its capacity to serve more concurrent users. Scalability refers to application's ability to scale with:

- No modification of code;
- No compromised data consistency;
- No deterioration of user-experienced latency beyond the acceptable level.

Relation between processing capacity and number of process instances of the scaled business process application is not linear. The processing capacity cannot grow in line with the increase of running process instances. Fundamental reason behind this is the fact that a complex distributed processing system consists not just of elements that perform in parallel – some of the elements perform sequentially. The non-linearity is source of two barriers related to economic and functional efficiency of scaling:

- **Marginal costs barrier** (infrastructure efficiency barrier). Costs of scaling increase faster than the achieved increase of processing capacity and reach a point where scaling beyond is not economically justified.
- **Marginal latency barrier**. With scaling, users-experienced latency increases faster than the increase of processing capacity and reaches a point where further scaling will make it higher than the critical level.

Non-Linearity of Scaling

Figure 1 illustrates a linear and a non-linear relation between the growth of process instances and resulting growth of processing capacity.

In theory, a linear scalability requires all individual processing units to share nothing. For example, each processing unit must run on a dedicated hardware, must have its own database, etc.

In practice, the nature of business processing necessitates the existence of a database that is common for all processing units.

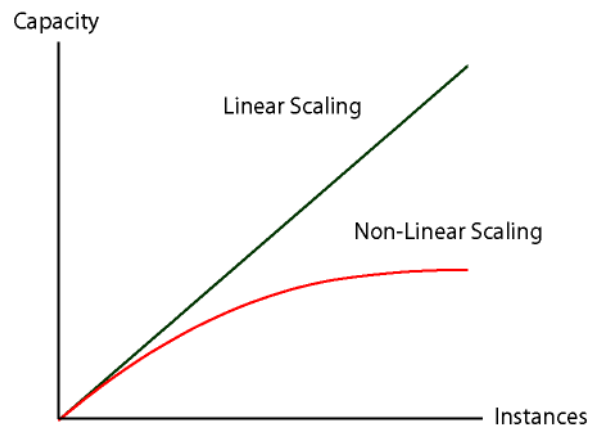


Figure 1 – Linear and Non-Linear Scaling

A near linear scalability might be achieved for a class of applications with **insert-only** data operations. In that case, each processing unit might update its own on-board database, while a background process is responsible for copying data from individual databases into a common database.

In general case, business processes require reading from and writing to a database that is common to all individual processing units. With increasing the number of performing in parallel process instances, database operations start becoming a performance bottleneck and cause non-linear increase of processing capacity.

Reason for Non-Linearity

Amdahl's law is used to predict the speeding up where the performance of only part of the system is improved. According to it, the speedup of a system with parallel and sequential fractions is limited by the time needed for its sequential fraction¹.

$$\text{Speedup} = 1 / (S + P/N)$$

Where: $S+P=1$,
 S represents the sequential fraction,
 P represents the parallel fraction,
 N represents the degree of parallelism.

For example, when $S=0.1$ and $N=50$, $\text{Speedup} = 1 / (0.1 + 0.9 / 50) = 8.475$

When $S=0.1$ the maximum achievable Speedup is 10, when $S=0.05$ the maximum Speedup is 20 etc, regardless of how large is the number N.

Marginal Costs Barrier

Figure 2 introduces the Acceptable Marginal Economic Efficiency of Scaling Line. It represents the acceptable proportion of process instances wasted in non-linear scaling. Point X is the point of intersection between the process' non-linearity curve and the Acceptable Marginal Efficiency Line.

Point A, the projection of point X on the horizontal axis, is the Marginal Costs barrier. The creation of process instances is backed by dedicated hardware and software licenses and this line represents the acceptable proportion of wasted in scaling infrastructure and its money equivalent.

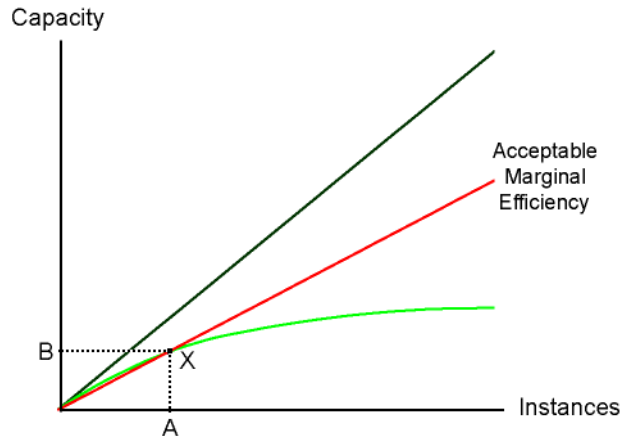


Figure 2 – Acceptable Marginal Economic Efficiency Line

The Marginal Costs barrier exists as costs of scaling increase because of the non-linearity and reach a point where scaling beyond is no more economically justified. Beyond this barrier, costs of scaling are no more acceptable as they continue getting higher.

Point B, the projection of point X on the vertical axis, represents the maximum processing capacity that might be reached with scaling that is perceived as economically efficient. Beyond this point, price for processing capacity growth is not acceptable.

¹ Gene Amdahl. "[Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities](#)". 1967. AFIPS Conference Proceedings (30): 483–485.

1.4. Marginal Latency Barrier

Figure 3 introduces the Max Acceptable Latency. Point C represents the Marginal Latency barrier. Beyond this barrier, users have to bear average latency that is higher than the Max Acceptable Latency users should experience.

Marginal Latency barrier is caused by the same reason that causes the scaling non-linearity. With scaling, users-experienced latency increases faster than the increase of processing capacity and reaches a point where further scaling will make it higher than the critical level of Max Acceptable Latency represented with the interval YZ.

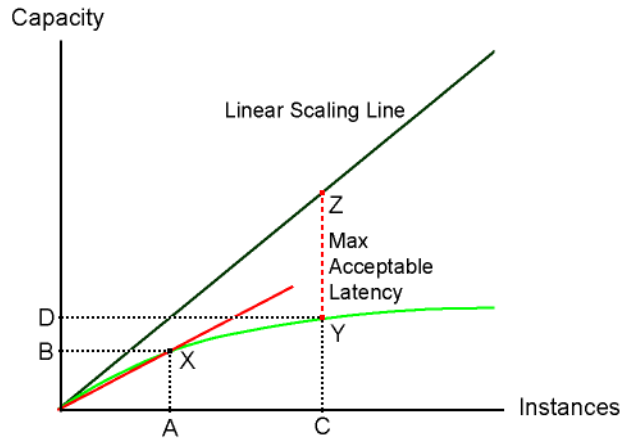


Figure 3 – Max Acceptable Latency

The Marginal Latency barrier refers to maximum acceptable increase of the average users-experienced latency per added infrastructure for a new concurrent user. It is a functional efficiency barrier. Beyond it, the system as a whole cannot provide the required quality of service and its functional efficiency is considered deteriorated below the acceptable limit. This deterioration is measured with the length of user-experience latency.

User-experienced latency has multiple components. With regard to the Marginal Latency Barrier, the important one is caused by the non-linearity of scaling. This component's value is proportional to size of the interval between scaling curve and the linear scaling line divided by the number of concurrent users the application can serve.

Scaling the process instances from point A to the Marginal Latency barrier represented by point C increase application capacity from point B to point D. Interval YZ represents the sum of non-linearity caused components of individual user latencies. Therefore, the value of non-linearity caused component of user-experienced latency is proportional to distance between point A and point Z divided by the value of point C.

2. Non-Linearity Curve

According to Amdahl's law, shape of a curve that represents the non-linearity of scaling depends on the proportion of parallel and sequential fractions of an application system.

2.1. Function of Data Processing Nature

Business processes require support of a database. With increase of the number of performing in parallel process instances, database operations start becoming a performance bottleneck, as the parallelism of database operations cannot be increased to match the number of process instances.

Figure 4 shows the scaling curves of three categories of business process applications. Database operations of each category reflect. The different nature of data processing in each category defines the speed and parallelism of database operations and shapes the steepness of non-linearity curves.

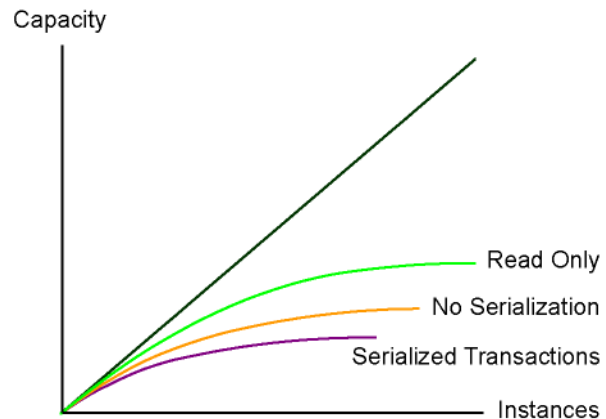


Figure 4 – Nature of Data Processing and Non-Linearity

Some processes perform only database reading. Other perform transactions with database writing that does not require serialization. Database writing of business processes that need to update one more aggregate values are performed within serialized transactions.

Data 'read' operations are generally faster than data 'write'. In addition data writing is generally performed within a transaction, which makes it even slower. This explains why even though both data 'read' and non-serialized data 'write' operations parallelize well, applications with data 'read' only operations experience less database caused bottlenecks and scale more efficiently than the other two types.

Data 'write' operations executed within serialized transactions are the major force behind database-caused bottlenecks in application performance. Serialization of short database transactions is a source of performance bottlenecks but not the worst. The most severe source of early appearance of performance bottlenecks are the serialized long transactions.

The only simple way to increase the performance parallelism of serialized long database transactions is their decomposition into a number of short transactions²

² More details on transaction decomposition for concurrent execution are presented in Transactional Cascade® Concept Paper Two: [Interactive Responsiveness and Concurrent Workflow](#)

2.2. Effect of Increased Parallelism

As forecasted with Amdahl's law, any increase in parallelism will push the scaling curve up, everything else being equal.

In respect to database operations, the increased parallelism will make the bottleneck to start appear at a later stage during the application scaling.

As it is shown on figure 5, the increased parallelism will in effect increase the maximum of achievable processing capacity.

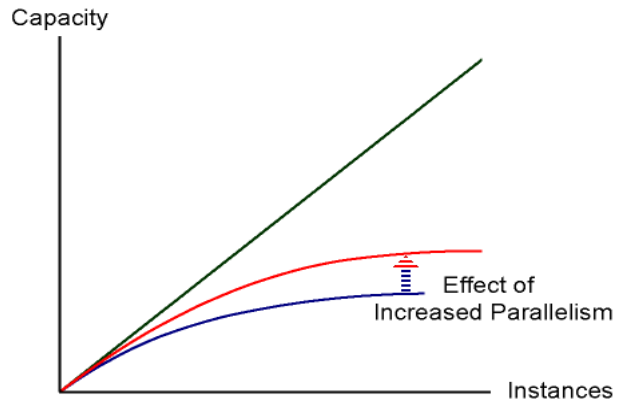


Figure 5 – Pushing the Non-Linearity Curve Up

3. Barriers in Different Application Categories

Steepness of the scaling curves is different not only between different application categories. It is different between applications within the same category and it is different even between different implementation of the same application.

3.1. Marginal Costs Barriers

Figure 6 presents intersections of scaling curves of different application categories with the Acceptable Marginal Economic Efficiency line. Projections of the intersection points on horizontal axis – point A, B, and C – represent the Marginal Costs barriers of scaling of different application categories.

Figure 6 might be used to visually explain the accomplishments in utilization of the Internet in some areas during the first decade of the new millennium and the non-accomplishments in other areas.

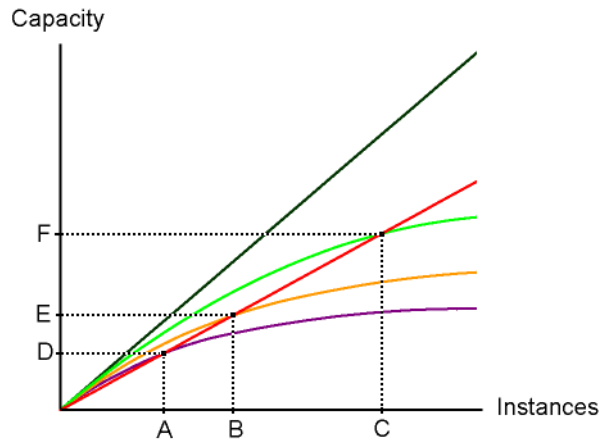


Figure 6 – Marginal Costs Barriers of Scaling

The biggest breakthrough is social networking. Social networking applications are dominated by data 'read' operations. Data 'write' operations are of 'insert' type and in considerably small proportion. Application responsiveness when data 'write' operations are performed is good but obviously it is performed on a secondary database with a slow replication on the main database by a background process.

Explanation: The majority of social networking user activities are performed by applications with Marginal Costs barrier presented on figure 6 by point C. These applications are cheapest to scale and they could get scaled to a high processing capacity. Social networking database writing is of smaller proportion. Part of the application that performs it has Marginal Costs barrier presented on figure 6 by point B.

Unlike normal business applications, social networking does not need to perform and large-scale data 'update' operations with aggregate values and consequently does not need large scaling of part of an application with a lot of long-executed serialized transactions. Therefore, the Marginal Costs barrier presented on figure 6 by point A has a little relevance and impact.

In resume: Social networking success with the global impact was possible because of:

- The highest proportion of processing capacity scaling is restricted by a barrier presented with point C. Here responsiveness is critically important
- The smaller proportion of processing capacity scaling is restricted by a barrier presented with point B. Here the responsiveness is artificial – users receive a response before the requested database writing is actually performed.
- A negligibly small (if any at all) proportion of processing capacity scaling is restricted by a barrier presented with point A.

In contrast to social networking, example of a non-accomplishment is the unsuccessful transformation of enterprise business process applications into Internet-scale applications. The reasons should already be obvious: The nature of various business processes requires serialization of relatively long database transactions. A barrier presented with point A restricts scaling of business processes with such type of transactions.

3.2. Marginal Latency Barriers

Figure 7 presents scaling curves of different application categories and each curve's Max Acceptable Latency interval – XT, YU, and ZV. Points A, C, and E are the Marginal Latency barriers of these curves.

This figure shows the correlation between steepness of scaling curve and scaling reserve of a process before its Marginal Latency barrier is hit. It illustrates the fact that steeper curves have higher reserves for scaling with regard to latency.

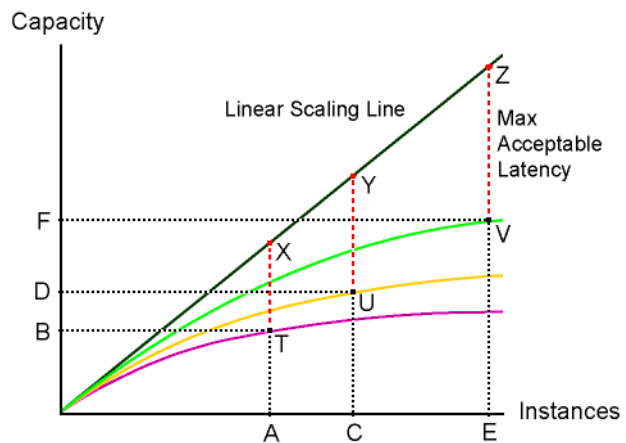


Figure 7 – Marginal Latency Barriers of Scaling

Key Point

Eventual future success in creation of Internet-scale business process applications depends on success of a middleware platform with ability to:

- Promote parallelism with long-running serialized database transactions
- Enable processing capacity scaling with no Marginal Costs barrier
- Enable scaling restricted only by Marginal Latency barrier

4. Middleware Architecture and the Barriers

Architecture of process middleware plays two equally important roles in application scaling. The first one is to provide technical means for increasing the upper limit of scaling of application processing capacity. The second one is to provide a software solution to the problem with low CPU utilization, existing where middleware with CPU-based parallelization is implemented, with a goal to eliminate the Marginal Costs barrier of scaling.

To increase the upper limit of processing capacity scaling, the middleware architecture should enable decomposition of a complex transaction into number of shorter sub-transactions. This should be coupled with means guaranteeing that all sub-transaction will either commit or the results of already committed sub-transactions will be rolled back with execution of compensators.

To increase the CPU utilization, the middleware architecture should enable creation and execution of high-volume of concurrent process instances on a single CPU core. Having this result achieved will eliminate the Marginal Costs barrier of scaling, as this barrier exists because the CPU-based parallelization did not have any alternative so far.

4.1. Workflow-Enabled Concurrency of Database Transactions

Concurrent performance of multiple instances of a serialized database transaction is a logical nonsense. However, the concurrent performance of instances of a long serialized transaction could become a fact with transaction chopping into short sub-transactions and execution of sub-transactions from a workflow system.

In this case, the workflow system must be able to provide 2 out of 4 main services expected from a transaction manager – atomicity (all or none of executed transaction members commits) and durability (transaction starts and ends with a persisted data, and transaction in progress survives a power outage).

With Transactional Cascade® workflow technology execution of an instance of a serialized long database transaction chopped into short sub-transactions, locks are acquired only for the duration of individual sub-transactions³.

³ More details on transaction chopping for concurrent execution are presented in Transactional Cascade® Concept Paper Two: [Interactive Responsiveness and Concurrent Workflow](#)

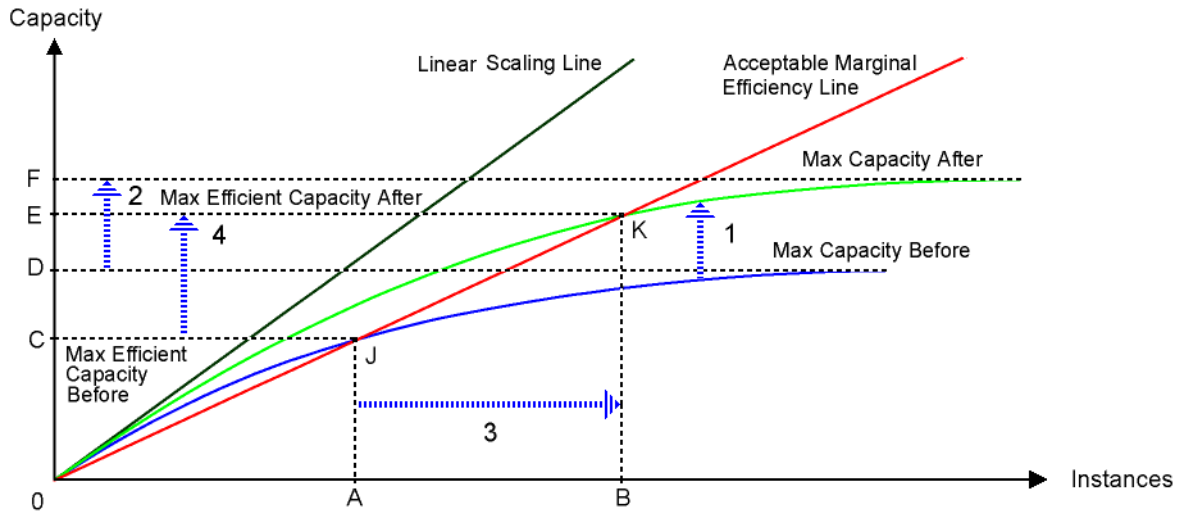


Figure 6 – Effects of Increased Parallelism of Database Transactions

Once concurrent performance of multiple instances of a serialized database transaction could become a fact, let see how it can improve the scaling of processing capacity of the notoriously difficult for scaling processes with serialized transactions.

According to Amdahl's law, the effect of increased parallelism of database transactions will show itself with pushing the scaling curve up, presented on figure 6 with number 1. Pushing the scaling curve up increases the maximum processing capacity. It is presented on figure 6 with number 2. Its value changes from being presented with point D to being presented with point F.

The new scaling curve causes an effect presented on figure 6 with number 3. The process' Marginal Costs (Infrastructure Efficiency) barrier initially being at point A moves to the right at point B. The new Marginal Costs barrier permits efficient scaling the number of process instances from being at point A to being at point B by adding new middleware infrastructure.

The effect presented with number 3 caused the effect presented with number 4 – an increase of the maximum efficient processing capacity from being at point C to being at point E. Scaling the number of process instances with adding new middleware infrastructure increases the quantity of running instances to point B, which increases processing capacity to a new value presented with point E.

Key Point

Workflow-enabled parallelism of database transactions increases the maximum achievable processing capacity and permits infrastructure scaling to a new Infrastructure Efficiency barrier, thereby increasing the available processing capacity.

4.2. CPU Utilization and Marginal Costs of Scaling

With CPU-based parallelization of the middleware, scaling requires additional middleware infrastructure – hardware and software licenses, more energy, higher infrastructure overhead, more floor space, support personnel etc.

With application server runtime capable of running high volume high volume of concurrent instances of a business process, thereby having high CPU utilization, the **marginal cost of scaling is zero**. For example, middleware infrastructure that runs a single instance of Transactional Cascade® workflow engine performs pure software scaling of application capacity from a level required for serving 1 user to a level required for serving 1000 users.

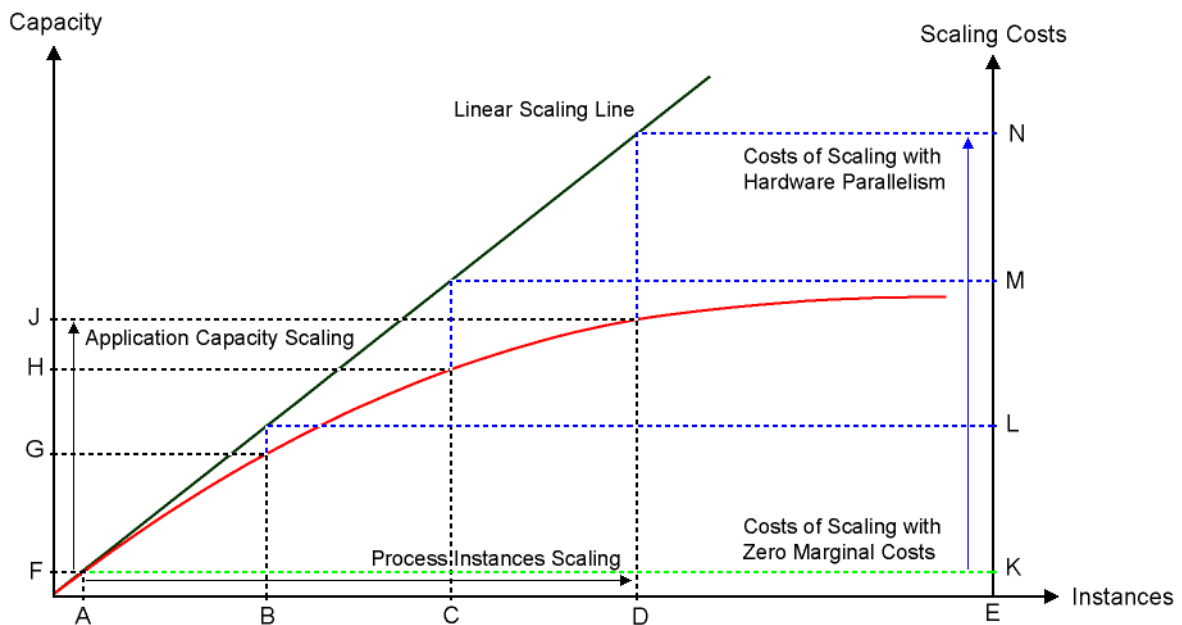


Figure 7 – Scaling with CPU-based Parallelism and Scaling with Software Concurrency

Figure 7 presents two types of costs dynamics related to principally different methods of scaling. With the first type, costs increase in direct proportion with increase of the number of process instances. With the second type, costs are constant and are not influenced by the increase of the number of process instances.

With both methods of scaling - CPU-based parallelism and software concurrency - the scaling curve is roughly the same as reason for its existence is the process performance bottleneck caused mainly by the inability to scale database operations in line with the scaling of business process instances.

Point A on figure 7 presents the number of process instances where both methods of scaling require the same quantity of infrastructure – hardware and software licenses. With the method of software concurrency, scaling the process instances from point A to point D, which increases the processing capacity from point F to point J does not require any additional costs – the total costs stay at point K.

In contrast, with the method of CPU-based parallelism, scaling the number of instance from point A to point B will increase the processing capacity from point F to point G and will increase the infrastructure-related costs from point K to point L. The same with the next two steps of scaling where the increase of processing capacity to point J increases costs to point N.

Key Point

When scaling with software concurrency, marginal costs of increased processing capacity are zero.

4.3. Scaling with No Barriers

Scaling with zero marginal costs might seem too good to be true. In fact it is just implementation of well-known methods of software design. The only new element associated with it is the challenge of the hardware efficiency of virtual execution engines when being implemented as server-side technology in general and as application servers' runtime in particular.

Ability to scale with no economic efficiency barrier, however, does not ensure that scaling to the absolute maximum is practically meaningful. Somewhere at the middle between the maximum efficient scaling with CPU-based parallelism and the absolute maximum of scaling another barrier will appear – the one of functional efficiency. When scaling beyond this barrier, the latency experienced by application users increases above the critical levels.

4.4. Scaling with No Marginal Costs Barrier

Transactional Cascade® workflow technology creates high volume of new business process instances with the same hardware and the same software licenses. As a result, marginal costs of scaling are zero. Consequently, scaling does not experience Marginal Costs barrier and it is not restricted by it.

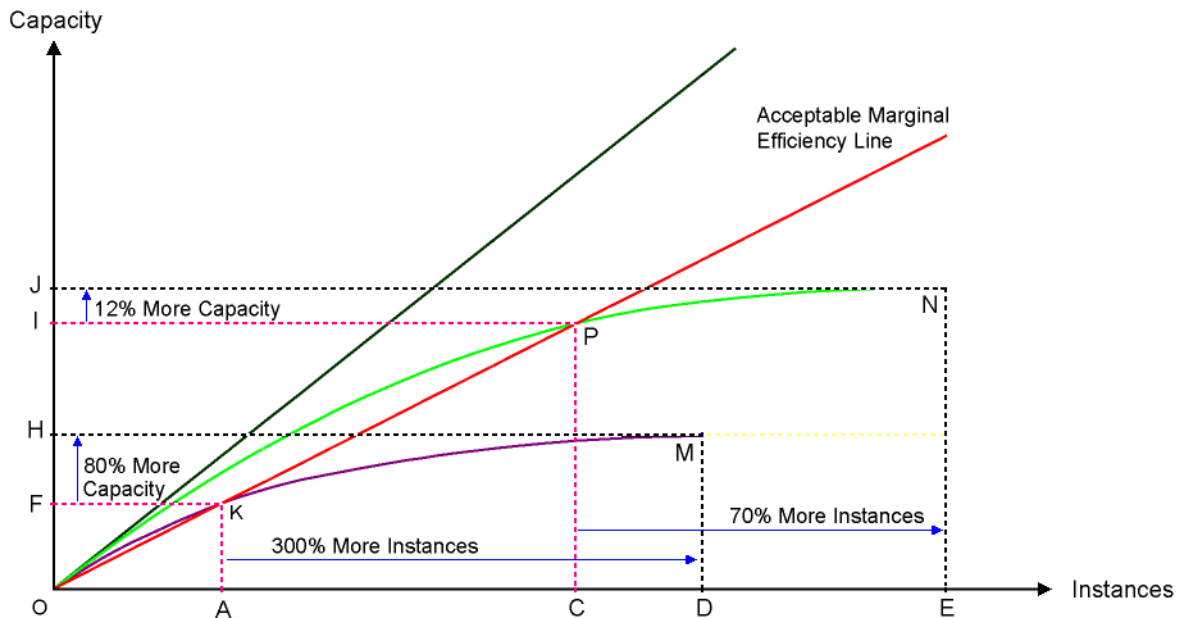


Figure 8 – Scaling with No Marginal Costs Is Limited by Latency Barrier

Figure 8 shows the scaling curves of two business processes. Point A and point C represent the maximum scaling with CPU-based parallelization of first process and second process as established by the Acceptable Marginal Efficiency line.

When the software concurrency method is implemented, scaling does not experience Marginal Costs barriers and both processes could be scaled to their absolute maximum. On the graphics of figure 8, switching from CPU-based parallelization to software concurrency permits increasing the running instances of first process with 300% - from point A to point D - and the running instances of second process with 70% - from point C to point E.

As a result of the switching to software concurrency, the maximum economically efficient capacity of the first process increases with 80% - from point F to point H - and the maximum economically efficient capacity of the second process increases with 12% - from point I to point J.

4.5. Scaling With a Marginal Latency Barrier

Functional efficiency barrier of business process scaling is the Marginal Latency barrier. Beyond this barrier, users have to bear average latency that is higher than the Maximum Acceptable Latency users should experience. Despite the real ability for economically efficient scaling to the absolute maximum, it rarely delivers a practical value because of the Marginal Latency barrier.

Marginal latency barrier is caused by the same reason that causes the scaling non-linearity. With scaling, users-experienced latency increases faster than the increase of processing capacity and reaches a point where further scaling will make it higher than the critical level of Maximum Acceptable Latency.

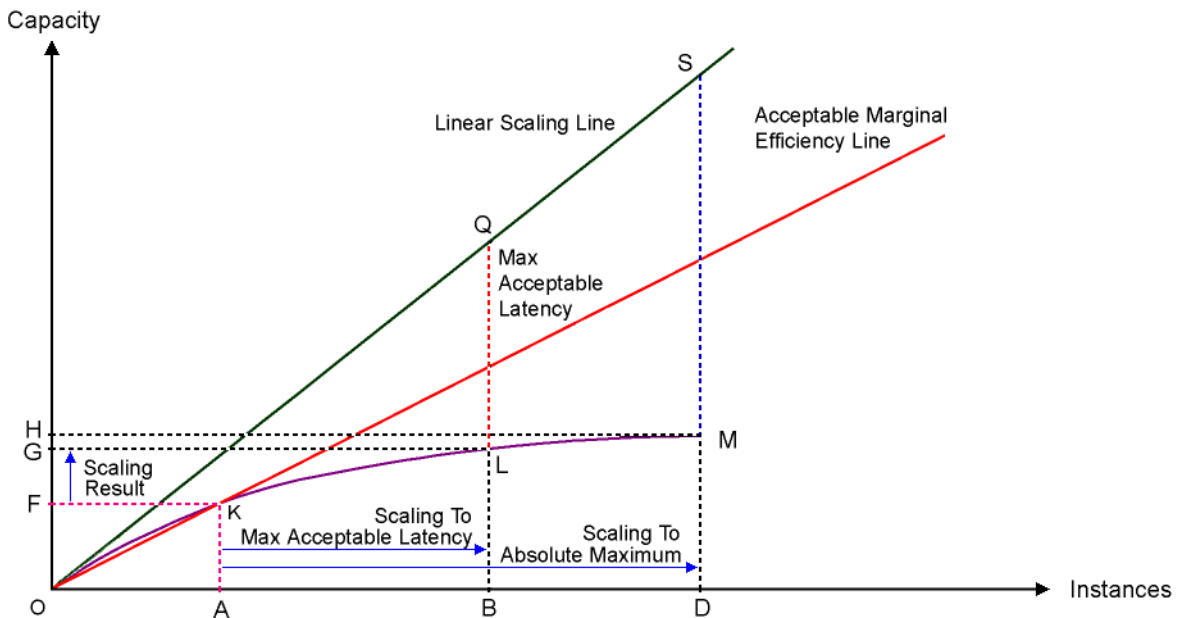


Figure 9 – Scaling and Marginal Latency Barrier

Scaling curve represents the increase of processing capacity that results from adding a new process instance. Number of process instances reflects the number of user interactions the application can process concurrently. The presented on figure 9 intervals QL and SM – the difference between the line representing linearly increased processing capacity and the curve of non-linearly increased processing capacity – represent the increase of process latency with scaling.

Marginal Latency barrier refers to maximum acceptable increase of the average users-experienced latency per added infrastructure for a new concurrent user. Apparently, scaling the application up to the point of absolute maximum D does not deliver any practical value, as the efficiency of the whole system will fall below the acceptable level of latency.

On figure 9, the Max Acceptable Latency is reached at the point B and this point represents the Marginal Latency barrier of scaling.

5. Conclusion

Transactional Cascade® workflow technology creates high volume of new business process instances with the same hardware and the same software licenses. As a result, marginal costs of scaling are zero. Consequently, scaling does not experience Marginal Costs barrier and it is not restricted by it. Scaling might continue up to the absolute maximum of achievable processing capacity, but there is no practical value associated with doing it. Scaling creates value when it is performed up to the Marginal Latency barrier.

Value Proposition

Compared with middleware capable only of CPU-based parallelism, scaling with Transactional Cascade® workflow technology creates value in two directions:

- It saves huge infrastructure budgets and energy costs.
- It enables scaling of a business process more than any other known technology – up to the point of maximum acceptable process latency.

[Give a feedback or ask questions](#)

TRANSACTIONUM PTY LTD

Tel: + 61 2 9567 1150

Fax: + 61 2 9567 1160

WWW.TRANSACTIONUM.COM

PO Box 324, Brighton-Le-Sands NSW 2216, Australia