A systems−performance equation for multi−thread systems

James Bradley
Department of Computer Science
University of Calgary,
Calgary, Alberta, Canada

Abstract A fundamental systems−performance equation that relates system throughput capacity to resources, resource−sharing procedures, and system threads, and which governs non−growth, agent−directed systems, is proposed and justified. The equation is called the *resource−sharing equation.* It shows how throughput capacity can be maintained by reducing resources and increasing resource−sharing procedure complexity, or vice versa.

All quantities used in the equation are precisely defined and their units specified. The equation reduces to a numerical expression, and can be subjected to experimental test. The equation clarifies and quantifies a basic principle, enabling designers and operators of systems to reason correctly about systems in complex situations. Spreng's Triangle, relating energy, time and information follows from the resource−sharing equation.

Key words *Complexity, performance, resources, sharing procedure, system, threads, throughput capacity.*

Introduction

Much of the complexity in computer systems is due to attempts to increase system throughput by increasing the utilization of underutilized but expensive system resources, the case of computer operating systems being one classic example [11], multiple−thread database management systems being another [4,5]. This increase in utilization is usually accomplished by sharing the resources among system threads [1]. Other sources of increasing complexity are due to efforts to apply procedures to reduce risk, particularly risk of deadlock or harmful interference between processes, the best known examples being critical section algorithms in operating systems [11]. This paper, however, does not deal with risk, and is restricted to considerations of the complexity resulting from the complex sharing procedures required to increase throughput capacity by sharing resources among system threads.

The systems to which these considerations apply are human−agent directed systems, particularly computer systems. Agent−directed systems are systems designed, constructed and operated by human agents. Unfortunately, the responsible agents may not always be able to reason clearly about the relationship between system throughput capacity, resources, and resource−sharing procedures and threads. As a result serious mistakes in system design and design strategy can occur [8,10].

In this paper we present a fundamental systems equation that relates system throughput capacity to resources and the resource−sharing procedures needed to share the resources among threads. It is one of five equations relating throughput and resources; the other four, involving system resources and risk, are not covered in this paper [3].

The research leading to this equation was inspired by phenomena in complex computer systems, and it is expected that it is on the computer systems arena that the equation will throw the most light. The equation does cover systems phenomena outside of the computer arena as well, particularly in industrial production systems [7], as will be exemplified near the end of the paper. Thus the equation expresses a unity of principle in a great diversity of system phenomena.

Mindful of the fact that most discussions of systems are weakened by lack of precise definitions and units of measurement, in this paper every attempt has been made to ensure that concepts are clearly defined, and that symbols appearing in the five equations denote quantities that are measurable and expressible in clearly defined units.

1.0 Overview

A system is considered to be any entity that converts inputs to outputs, and which may be composed of subsystems, connected either in series or in parallel, each of which is also a system. We also consider a system to be an entity that functions under the direction of one or more human agents, and which employs essential resources R, often portable, enabling it to convert a set of inputs N, informational or material or both, to a set of outputs U per unit time, so that U is the system throughput. If the maximum value for U is I, then I is said to be the throughput capacity of the system, measured as the units of output per unit time when the system is operating at maximum capacity. In general, if the system is operating at a fraction f of throughput capacity I, then the throughput is U = fI.

In some cases there will be more than one type of throughput product. In that case we can declare throughput capacity of one of them as being the main throughput product capacity I, in units of product per time period. It can then be assumed that each other product throughput capacity is a byproduct capacity that is a function of I, in most cases a linear function. Accordingly, in this paper, we deal only with I, considered to be either the only or the main throughput product capacity.

Overview of the resource sharing equations

A reference summary of the semantics for the resource sharing equation for throughput capacity I is given below. Minimal explanation is included here.

The resource sharing equation is

$$I = KR(1 - T_S/T)[1 + sF_T(T_S)] \qquad (1)$$

Independent variables under the control of the agent are R and T_S. K is a constant and R measures resources available to the system, alterable only in valid units of the type comprising R. T is a constant and measures the time for which I is computed. T_S has units of time per (basic harmonic) resource unit, and measures the execution time, and thus resource−sharing complexity, of a *normally complex, coordinated, resource−sharing procedure* for sharing resources within R among threads of throughput. $F_T(T_S)$ is a growth function that has value 0 when T_S is zero and increases at an decreasing rate with increasing T_S to saturate at 1.0. The constant s is the *available sharing−enhancement potential*, where sR is the effective increase in R due to execution of a coordinated resource−sharing procedure sufficiently comprehensive to saturate $F_T(T_S)$ at 1.0. T_S/T is a measure of the resource capacity diverted from normal operations to carrying out the resource sharing procedure, and is thus a measure of the negative impact of the sharing procedure on throughput capacity I. When there is no resource sharing, T_S and $F_T(T_S)$ are zero, and the expression reduces to I = KR. There is a value for T_S at which I is maximized, found by solving $dI/dT_S = 0$. When T_S approaches T and I approaches 0, the systems is said to be thrashing [10, 11]; most resources are devoted to resource sharing among the threads, and little useful throughput results.

Measure of resources

The units of both R and P are units of anything of value to a human agent. Thus R could be measured in dollars, marks, printers or printer equivalents, microprocessors or microprocessor equivalents, or even, where the resources are mostly human, such persons as programmers. Also, since a resource can be valued by the human effort or work, or work equivalent, required to produce it, and since work is a measure of energy, then ultimately resources R can be measured in units of energy [13, 14]].

2. The resource sharing equation

In this section we demonstrate the veracity of the resource sharing expression:

$$I = KR(1 - T_S/T)[1+ sF_T(T_S)] \qquad (1)$$
$$I = KR \qquad (1a) \quad \text{if } T_S = 0$$

When no coordinated resource−sharing procedure is involved, coordinated resource−sharing time T_S is zero, making $F_T(T_S)$ zero, so that expression (1) simplifies to I = KR. This expression states that, if we increase resources R (in a valid manner), then I will increase linearly with R. For example, if we construct a system that is an m−fold replica of the original system, with resources mR, then throughput capacity will be mI. We take I = KR as axiomatic for valid changes to R.

The basic linear relationship between throughput capacity and resources

Suppose we construct the smallest possible miniaturization of the original system with resources R/n and throughput I/n, such that, when this smallest miniaturization system is replicated n−fold, we recover the original system with resources R and throughput I. We call this smallest possible miniaturization of the system the basic harmonic (system) of the original system. In addition, we call the original system the nth harmonic (replication) system. Hence, for an nth harmonic system with resources R, the only valid increase or decreases in R in equation (1a) can be in multiples of R/n of the same type as R, or, more loosely, in basic harmonic resource units of the type already constituting R. We also call the resources R/n the *basic harmonic resource unit* of the system.

For a simple system consisting of 10 processors producing 30 x−units per unit time, increasing R by 5 similar processors will increase x−unit throughput capacity by 15 x−units; the basic harmonic resource unit is 1 processor, so that R can be changed in units of processors. However, changing R by adding resource units of a type different or not equivalent to those already constituting R (e.g. by adding 5 processors each capable of generating 6 x−units per unit time) is invalid as far as (1a) is concerned.

The importance of expressions (1a) holding only for R being alterable in terms of basic harmonic resource units of the type already constituting R can be can perhaps be more forcefully illustrated by the following obvious example. Suppose a software house with 4 programming teams, each team being 2 cooperating programmers, so that R = 8, when measured in programmers. Suppose also that each team functions independently to produce x average application programs per year, for throughput capacity I of 4x programs for the house. Thus the basic harmonic resource unit is a team of 2 programmers. If we add 2 independent programmer pairs or teams to the original 4, so that R = 12, I goes to 6x; the increase was valid, being 2 basic harmonic resource units. But suppose we add 4 programmers by adding one programmer to each of the former teams, so that we have 4 new teams, each of 3 cooperating programmers; although R now also measures 12, this is not an allowable alteration of R, for we do not have an increase in independent resource units of the type already in R, that is, an increase in the number of basic harmonic resource units. Instead we have created a new system, and in all likelihood, in this case, new capacity I will not be 6x, but less, or maybe even I < 4x!

Parallel and serial subsystems

The restriction on the valid variability of R, in I = KR, to changes in harmonic resource units of the same type as R, has significant consequences with composite systems consisting of either parallel or serial subsystems. We consider parallel subsystems first.

A. Parallel operation

Suppose a system consists of two types of resources, say i low capacity processors amounting to resources R_1 (e.g. 15 slow processors that each generate 3 x–units per unit time per processor) and j high–capacity processors amounting to resources R_2 (e.g. 10 fast processors that each generate 6 x–units per unit time per processor) with i > j. [We are not interested in economics here, just in throughput capacity – the upkeep cost of the slow processors may be less than half that of the fast processors, and so be more economic.] Suppose R_1 and R_2 operate in parallel, generating throughput I_1 x–units per time unit and I_2 x–units per time unit respectively, where the number of R_1 units can be increased or decreased independently of the number of R_2 units, and vice versa. In such a case the expression

$$I = K(R_1 + R_2) \qquad (2a)$$

will not hold for independent variability of R_2 and R_2. We have essentially two separate systems, or parallel subsystems, so that, for the composite system, $I_1 = K_1R_1$ and $I_2 = K_2R_2$, and

$$I = I_1 + I_2 = K_1R_1 + K_2R_2 \qquad (2b)$$

[= $3R_1 + 6R_2$ x–units per unit time, in the processor example]

Here R_1 is alterable in one set of independent units (slow processors), and R_2 in another (fast processors).

The expression $I = K(R_1 + R_2)$ will hold only if, when R_2 is increased or decreased by R_2/n units, R_1 is increased or decreased by R_1/n units where n is the largest common divisor of R_1 and R_2. In other words, we must increase in sets of R_1 and R_2 units (that is, in basic harmonic resource units), such as sets of 3 slow and 2 fast processors, for $I = (21/5)(R_1 + R_2)$, giving an nth harmonic system for n = 5.

[If n is very small, for example, n = 1 with 10 slow and 7 fast processors, we may prefer to perform a simplifying adjustment to the system. In the interests of a tidier, or more finely–grained, harmonic system, we could either add 4 slow processors (n = 7) or 3 fast processors (n = 10).]

B. Serial operation

If two subsystems each with resources R_1 and R_2 are operating in series, everything depends on whether of not both subsystems are operating at capacity, and to what extent the subsystem resources can be shared.

If both subsystems are operating at capacity, then throughput capacity $I = K(R_1 + R_2)$ holds. If it is a nth harmonic system, a valid increase is an increase in units of $(R_1 + R_2)/n$. However, if R_2 is operating below capacity, and R_1 is at capacity, then R_1 is the *limiting (or "bottleneck") resource*, and R_2 is the *non–limiting resource*. Although $I = K(R_1 + R_2)$ holds, it is also the case that $I = K_aR_1$ holds for increases in R_1 up to the point where R_2 starts to operate at full capacity, at which point $I = K_b(R_1 + R_2)$ holds for further resource increases in basic harmonic resource units.

Coordinated and non–coordinated sharing of limiting resources

Suppose again that two subsystems each with resources R_1 and R_2 are operating in series. If one of the resources R1 is limiting then R1 may be capable of participating in either *uncoordinated exclusive–allocation sharing* of resources among non–threads or *coordinated inclusive–allocation sharing* of resources among threads. In both cases there is sharing, but in each case the nature of the sharing is very different, and the difference is vital for understanding both equation (1) and the operation of complex systems. Rather than define the two resource–sharing concepts above at this stage an example is given below, to promote the reader's understanding of the differences between them.

Computer Operating System Example. Consider a computer system with a fast processor/memory unit (R_1) and 2 slow input–output device pairs (R_2). Each I/O device pair is at different user locations, and is in use for 10 minutes in each hour, and during the remaining 50 minutes of the hour new data is being prepared for entry, although the user still retains control of the I/O device pair.

During use of an I/O pair, input data (i.e., a "jobload" of data) is interactively entered and converted via processing to information output on the output device. During a 10 minute session, the processor is exclusively allocated to a specific I/O pair. Clearly, system throughput capacity is 2 jobloads per hour.

It is clear that R_2 is limiting. As we increase the number of I/O device pairs, capacity increases, in accordance with $I = KR_2$, to 4 jobloads per hour for 4 I/O device pairs, until finally with 6 I/O pairs capacity is 6 per hour. At this point the cpu/memory unit R_1 is being shared in an uncoordinated manner among the jobloads, with exclusive allocation of R_1 to each jobload, that is, a jobload must finish being processed before the processing of a new one can begin. Furthermore there is no sharing of R1 among threads of production, that make up each jobload.

If we now double the number of I/O device pairs to 12, further throughput capacity increase will not occur, but either 6 device I/O pairs will never be in use, or each I/O pair will be operating at half capacity, once every second hour, assuming we retain exclusive allocation of R_1 to a jobload.R_1 would now be limiting.

Alternatively, suppose the processing of a jobload, as is usually the case, consists of short cpu processing bursts. Suppose that each burst lasts 60 milliseconds on average, and 100 bursts are needed to process a jobload, so that 6 seconds of actual processing time are needed per jobload of data. In other words, when the sum of the operations needed to process a jobload of data are laid out in time, we have a thread, consisting of bursts of cpu processing of the jobload separated in time.

In that case, by interleaving the cpu processing of different jobloads [1], that is, by coordinated sharing of R_1 inclusively among multiple jobloads, we could process up to 594 extra jobloads per hour on top of the 6 that renders R_1 limiting. This increase is spectacular (because of the enormous cpu speed), compared to the increase in throughput capacity that can normally be obtained by coordinated sharing with inclusive allocation with other types of systems. At any instant the (limiting) processor/memory unit R_1 is being shared among the processing of more than one jobload (actually among 100 jobloads).

This type of sharing may be looked upon as interleaving the processing of the processing threads for each jobload, where the cpu processes a sequence of thread sections, with the individual sections coming from a variety of different threads.

In practice, given the numbers above, somewhat less than an extra throughput of 594 jobloads per hour will be achieved by the coordinated sharing, since some of the processing time available to R_1 will be devoted to coordinated sharing activities (processor scheduling, context switching, and dispatching etc.[6, 11]) during a time T_S. Equation 1 applies to this situation.

Units of the rate constant K and throughput capacity I

In the expression I = KR, the rate constant K can be either intradenominational or interdenominational. K is interdenominational if it is measured in x–units per y–unit, where x and y are different [7]. If x and y are the same units K is intradenominational. Intradenominational units occur mostly in financial systems. [For example, the return (I) at a rate K of 0.05 dollars per annum per dollar of R invested, or 5%.]

With computer systems K is nearly always interdenominational, such as jobs per minute per processor, or bytes per second per disk−drive controller. Such units are in common everyday use in the practical business of data processing. Note however, that it is possible to have intradenominational units for K if we measure R in dollars or joules, and I in dollars or joules per time period, so that K is dollars per dollar per time period or joules per joule per time period [13,14]. This will be relevant later, when Spreng's triangle [12] is considered.

In general, K is measured in units of output per time period per harmonic resources unit. In the simpler cases a harmonic resource unit is atomic, such as 1 processor, or 1 controller. However, in more complex cases a harmonic resource unit will be composite, made up of combinations of units of different resources, so that we can have K expressed as jobs per hour per (harmonic resources) set of 2 processor, 1 printer and 2 disk drives, or 1 operator and 3 computers. However, in cases where such an obviously awkward unit is nevertheless the correct one, an improvement may be to use the aggregate of a common attribute type, such as dollar value.

Throughput capacity I can be measured in throughput entity or throughput entity set units, or in throughput entity attribute units. When we use throughput entity units we use the number of entities throughput per time period, e.g. computer jobs per second, files per hour, pages per second, and so on. With entity set units, we simply use named sets of entities throughput per time period, e.g. diskloads of files per day, directories of files per hour, and so on.

The concept of coordinated sharing of a resource

The entity (or entity set) per time period measure of throughput enables a concise definition of coordinated sharing of a resource:

Resources R of a system or subsystem are being shared in a coordinated manner if, and only if, at any instant, more than one entity or entity set unit of system throughput is under processing by resources R, requiring that the R resources be engaged in (usually complex) coordination procedures for measurable periods of time.

Alternatively in terms of threads:

Resources R of a system or subsystem are being shared in a coordinated manner if, and only if, R processes the individual threads (corresponding to production entities) in thread sections, where processing of sections from different thread is interleaved in time, requiring that the R resources be engaged in (usually complex) procedures for coordinating the processing of the interleaved thread sections for measurable periods of time.

Thus a computer processor or human is participating in coordinated sharing if it (he/she) has the processing of more than one job or process under way at any instant.

Relationship between I, R and coordinated sharing time (or complexity) T_S

Consider any basic system of throughput capacity I and resources R, of arbitrary complexity, where some subsystem resources are connected in series and others in parallel, and some are shareable and some not. It follows from the discussion earlier that when the system is operating at capacity, some subsystems will be at capacity and others below capacity.

We can define an *available coordinated sharing−enhancement potential* s per harmonic resource unit of the system, where s > 0, such that resources R are effectively increased by sR, and thus throughput capacity I is increased by sKR, by means of coordinated resource−sharing up to the physical limit possible, so that after such resource sharing, I = KR(1+s). The value for s for a given system can only be obtained by analysis of the system. For example, in the case of the computer system earlier with the processor/memory unit limiting, R can be taken as 1, and K as 6, so that $I = KR = 6*1$ with no coordinated sharing. In theory, with coordinated sharing up to the limit possible, we have $I = 6R(1 + 594/6) = 100$ jobloads per hour, so that $s = 594/6 = 99$.

If only a fraction F of the available sharing−enhancement potential s is made available through limited coordinated sharing, I will be less than its potential $KR(1 + s)$ and will be given by $I = KR(1 + sF)$. F might be called the *available sharing−enhancement potential fraction*.

With any system, coordinated resource sharing among throughput entities requires that system resources R participate in a usually complex and time−consuming *coordinated resource−sharing procedure* to enable concurrent handling of multiple throughput entities without collisions. An important property of the sharing procedure is that is it requires allocation of resources R being shared for total time T_S of operation of the sharing procedure (per basic harmonic resource unit), thus temporarily diverting the resources from normal throughput generation. System agents, while seeking the increased throughput capacity benefits of resource sharing, will be inclined to seek to minimize system time T_S per harmonic resources unit lost to operation of the sharing procedure.

We now assert, as axiomatic, that, generally, system agents will undertake coordinated resource−sharing in steps, where the most effective, in terms of most throughput capacity increase for least sharing time T_S, is undertaken first, followed by the next most effective, and so on. Hence, as the fraction F of the available sharing−enhancement potential s, due to operation of a coordinated sharing procedure, increases towards 100%, each increased 1% gain in F requires a greater increase in coordinated resource−sharing procedure activity time T_S than for the previous 1%. This means that the available sharing−enhancement potential fraction F must be a function of T_S, that is $F = F_T(T_S)$. Empirically, $F_T(T_S)$ must have value 0 when T_S is zero and must increase at a decreasing rate with increasing T_S, to saturate at value 1.0.

Hence, if we neglect the costs of resource sharing, due to resource diversion for time T_S during operation of the sharing procedure, the relationship between coordinated resource−sharing procedure time T_S per harmonic resource unit and throughput capacity I must be given by an expression

$$I = KR(1 + sF_T(T_S))$$

where $F_T(x)$ is a growth function of the general form:

$$G(x) = (1 - e^{(-x/g)})$$

The constant g can be very small, so that G(x) quickly becomes 1.0 when x is only slightly greater than 0, corresponding to the case where very little resource sharing time T_S is needed to achieve 100% of available sharing−enhancement potential s. Note, however, that since $F_T(T_S)$ is essentially an empirical expression that will represent some average of a large number of similar functions for different situations, some departure from the ideal G(x) shape can be expected in any practical situation, although in all cases it will increase on average from 0 with increasing T_S to saturate near 1.0. Since T_S measures the time for which the sharing procedure is in operation, per basic harmonic resources unit, it may also be regarded as a measure of the complexity of the coordinated sharing operation, and thus the order created [31] by the sharing operation, and hence the decrease in entropy of the system.

In the above expression we have neglected the negative impact on I of operating the coordinated resource−sharing procedure during time T_S. If T is the period during which throughput capacity is measured, for example T = 1 hour, where I is in jobloads per hour, or T = 1 day, where I is diskloads per day, then the fraction of the time during which R is diverted from normal throughput is T_S/T. Hence operation of the sharing procedure effectively reduces R to $R(1 - T_S/T)$. Hence the correct expression for the relationship between I, R and T_S must be

$$I = KR(1 - T_S/T)[1 + sF_T(T_S)] \qquad (1)$$

which is the resource sharing equation.

Equation (1) is stating that following introduction of a coordinated resource sharing procedure taking time T_S per basic harmonic unit of resources, with $F_T(T_S)$ anywhere between 0 and 1.0, and consequent increase in I is as given in (1), it is then possible to increase(decrease) R in basic harmonic units subjected to the same level of sharing, and obtain a further increase(decrease) in I. However, R has to be increased in valid units, which means in units of resources equal to the smallest functional miniaturization of the system resources, or basic harmonic resources units. Also the increases in R must be shared in the same way as exiting units of R. It is important to understand that T_S is the time taken for sharing the resources in such a minimum unit of resources, and is thus independent of the magnitude of R. The correct units of T_S are therefore units of sharing time per unit of basic harmonic resources. Of course, after some time, if the system is operated with a fixed or standard level of sharing, then $(1 – T_S /T)[1+ sF_T(T_S)]$ is constant, and can be absorbed into the rate constant K as K_S in $I = K_SR$, the resources of any increase in R being assumed to be shared in the standard manner.

It can happen that too much of the extra throughput benefits from sharing resources may be offset by the cost of the sharing, so that the criterion for productive use of a sharing procedure is obviously:

$$(1 – T_S/T)[1+ sF_T(T_S)] \quad > 1$$

Where this inequality does not hold, coordinated sharing cannot pay.

If equation 1 is analyzed carefully, and examples pondered, it will be seen that there is a level of sharing for which I will be a maximum. If T_S is small relative to T when $N(T_S)$ approaches 1.0, then this maximum will be

$$I = KR(1 – T_S/T)[1+ s]$$

This reduces to $I = KR[1+ s]$ if T_S is really small compared to T as $N(T_S)$ approaches saturation.

At the opposite extreme, T_S could approach T before $N(T_S)$ reaches saturation, at which point I will approach zero. Thus in the general case I will increase with increasing T_S to a maximum value, and then decline, possibly to zero, as T_S continues to increase. A declining I for increasing T_S is the phenomenon of thrashing, where increasing the level of coordinated sharing results in less throughput capacity because the incremental cost of operating the sharing procedure exceeds the incremental benefit. Thrashing is a well–known phenomenon in computer operating systems [29], but is possible in any system with coordinated sharing. If we set $x = T_S$, the peak value for I is obtainable from a solution of

$$dI/dx \;=\; – KR/T[1+ s(1 – e^{(–x/g)})] \;+ KR(1 – x/T)(1/g)se^{(–x/g)} = 0$$

which, since it involves standard calculus, is left to the reader.

It should now be clear that the increased throughput capacity caused by coordinated sharing of resources within a system always has to be "paid for" by means of resource–consuming complex coordinated resource–sharing procedure activity, in which the complexity may be measured by T_S, and this axiomatic rule cannot be evaded. Readers who merely look around, will see it in operation in almost every agent–directed system, whether simple or complex, whether technological or social

Spreng's Triangle

D. T. Spreng, a physicist, hypothesized that for any given task, for example to produce output q,

$$f_1(t)f_2(E)f_3(i) = \text{a constant}$$

where t is the time, E is the energy, and i is the information required to perform the task, and where $f_1(x)$, $f_2(x)$, $f_3(x)$ are functions whose value generally increases with increasing positive x. This is one statement of what has come to be known as Spreng's Triangle [2, 12, 13, 14]. In other words to carry out a given task, you can save energy by taking more time or using more information, or you can save time by using more energy, and so on.

Spreng's triangle can be deduced from the resource sharing equation:

$$I = KR(1 - T_S/T)[1+ sF_T(T_S)] \qquad (1)$$

Suppose we need to generate a quantity of output q. If this is done in time t with the system operating at maximum capacity then $q/t = I$ and:

$$q = KtR(1 - T_S/T)[1+ sF_T(T_S)]$$

In this expression, we can let q represent the fixed task, so that

$$tR(1 - T_S/T)[1+ sF_T(T_S)] = \text{a constant}$$

But R is a measure of the energy required in joules, so that $R = f_2(E)$, and T_S is a measure of the order in the sharing procedure, hence a measure of negative entropy, hence a measure of information. Hence $(1 - T_S/T)[1+ sF_T(T_S)]$ can be written as $f(T_S)$, which can be written as $f_3(i)$ where i is information. Hence:

$$tf_2(E)f_3(i) = \text{a constant}$$

It follows that the sharing equation is stating some fundamental limits [2] about the nature of physical reality.

Resource R adjustments

As we have seen, the increase in throughput capacity due to coordinated sharing is caused by the resources of at least one limiting subsystem being shared among the throughput entities feeding to or from underutilized subsystems. However, as a result of the sharing, sometimes adjustments to R, not covered by equation (1), are desirable. [Note that this subsection on adjustments is included for purposes of completeness, and may be skipped on initial reading.]

Take a system consisting of a computer processor/memory unit with many input and output device pairs with no coordinated sharing of the processor. A jobload of data coming from an input device (in continual disparate input bursts) and going to the output device of the I/O device pair (also in continual disparate bursts) has the processor exclusively allocated until the jobload of data has been completely output. The processor/memory unit is the limiting resource and so many I/O device pairs are underutilized. But if the processor can have multiple jobloads under processing at once, with the processing of the bursts of one jobload being interleaved among the bursts from another job, we can have sharing of the limiting resource. Because of higher throughput capacity, consistent with (1), this will have the effect of converting the I/O devices from underutilized resources to better or fully utilized resources. Following sharing of limiting resources to the limit possible, there are essentially two distinct possibilities for adjusting R by adding further subsystem resources to allow for better subsystem matching.

Case 1. Adjustment with unsharable resources R_u Suppose that the computer system with resources R has 1 processor (with sufficient memory) and 10 I/O device pairs, each pair being used for a single job or jobload of data. Now suppose that the processor is shared, in an inclusive coordinated manner, among the jobloads from all of the I/O device sets, so that $F(T_S) = 1.0$, but the situation is such that the processor could be shared among the jobloads from12 I/O device pairs as a maximum. In that case an even greater sharing−enhancement of R is possible, that is, a larger s is possible. An adjustment of 2 sets of I/O−device unsharable resources R_u could be added to R, for a total of 12, such that if the sharing procedure and thus T_S is extended to cope with them, additional throughput will result, with a new version of (1) being applicable:

$$I = K_u (R + R_u)(1 + T_S/T)[1+ s_uF_{Tu}(T_S)]$$

Addition of R_u to R is not a valid increase in R in (1), and causes the original system to be fundamentally changed with respect to resources, so that modified constants K and s, and function $F_T(T_S)$ are now required.

Prior to undertaking additional sharing to cope with the addition of R_u, $F_{Tu}(T_S)$ will be less than 1.0. Adding appropriately to T_S will bring $F_{Tu}(T_S)$ to saturation at 1.0, thus completing sharing to the maximum possible allowed for by the nature of the processor. At this point resources $R + R_u$ may be said to constitute not only a basic harmonic resources unit, but a sharing–potential maximized basic harmonic resource unit. and from this point resources can be increased in units of $R + R_u$ with maximum efficiency resulting, that is, maximum increase in I per resources unit.

Case 2. Adjustments with sharable resources R_S Suppose that the computer system with resources R has 24 I/O device pairs, each pair being used for a single job or jobload of data. Now suppose that, because of the nature of the processor it can be shared among the jobloads from 12 I/O device sets at a maximum, so that $F(T_S) = 1.0$. This means there are 12 extra I/O device pairs than cannot be used efficiently and which are contributing nothing to capacity. But this does not mean that they can be removed from the system without effect, for they can still be in use! Every pair is likely to have users, and which pairs are regarded as superfluous as far as throughput capacity is concerned is arbitrary, for we can simply regard all the I/O device sets as being one half underutilized.

To deal with this situation by improving throughput capacity, we could adjust R by adding 1 processor (sharable resources R_S) to give 2 processors and 24 sets of I/O devices, so that a new version of (1) is applicable:

$$I = K_g (R + R_S)(1 + T_S/T)[1+ s_g F_T(T_S)]$$

Addition of R_S to R is not a valid increase in R in (1), and causes the original system to be fundamentally changed with respect to resources, so that modified constants K and s, but this time no modified function $F_T(T_S)$, are now required.

Prior to the adjustment, the sharing time T_S was for 1 processor being shared among the equivalent of jobloads from 12 sets of I/O devices. Following the adjustment, the basic harmonic resources unit is 1 processor and 12 sets of I/O devices, so that exactly the same sharing time T_S per basic harmonic resource unit is required, and thus the same function $F_T(T_S)$. At this point resources $(R + R_S)/2$, or 1 processor and 12 I/O device sets, are the basic harmonic resources unit, and also the sharing–potential maximized basic harmonic resource unit. From this point resources can be increased in units of $R + R_S$ with maximum efficiency resulting, that is, maximum increase in I per resources unit.

Note however, with R being 2 processors and 24 sets of I/O devices, if we continue to adjust R by adding processors, which are sharable resources, only minor and possibly no increased throughput capacity results. The level of coordinated sharing, however, as measured by the value $F_T(T_S)$, must fall, since we are increasing the number of processors per I/O device set, until when there are 24 processors there is no sharing at all, $F_T(T_S)$ is zero, and each set of I/O devices has its own processor.

Risk of capacity loss due to deadlocks and collisions

A second–order effect due to the use of a coordinated sharing procedure is that sometimes risk of throughput capacity loss appears as a side effect, for example, due to deadlocks, and critical–section (interference or collisions) problems in computer operating systems [6, 11]. In general, such risks are the consequence of mismatches between system resources for a given sharing procedure and input stream, and will be considered elsewhere [3] in terms of equations dealing with the relationship between throughput capacity, resources and risk.

4. Resource sharing in complex manufacturing systems

As is well known, many complex software products, besides operating systems, manage throughput by means sharing resources among threads. A common example is a database management system, where there is a thread associated with each request for service, either for data retrieval, data insertion or data update [4,5]. Equation 1 applied to all such systems.

The resource sharing equation also applies to physical systems, where the inputs and outputs are material, for example production lines in a manufacturing plant [7]. Throughput capacity can often be increased in such systems by coordinated sharing of resources, among many production threads, but only at the expense of a complex coordinated resource−sharing procedure. There is no escaping the consequences of the sharing equation. Most of the examples here involved specialized knowledge though, beyond the scope if this paper [9]

However, there is an obvious example that is instructive. Consider an automotive production system, and the paint facility in particular. Suppose each automobile needs three coats of paint. Suppose it takes 1 minute for a computer−controlled paint−sprayer to spray on a coat, and 9 minutes for it to dry, before the next coat can be applied. In that case the facility can produce 1 fully painted vehicle per half−hour, or 48 per day. If the facility is considered as $R = 1$, then $I = 48R = 48$.

But the facility is inactive 9 minutes out of ten, so that it is possible to share the facility among up to 10 production threads, or 10 vehicles. While the paint of one is drying for 9 minutes, coats can be applied to 9 other vehicles, each vehicle being considered as a thread of work with three separate thread sections, one for each coat. The resource sharing equation (1) will therefore apply.

If, ideally, $T_S = 0$ and $H(T_S) = 1$, we will now have $I = 48R(1 + 9)$, so that $s = 9$, and throughput capacity will jump from 48 per day to 480 per day. In practice, T_S will be greater than 0 and $H(T_S)$ will be less than one at maximum throughput capacity, since the facility will consume some time in a coordinated procedure needed to share itself among the threads of production. Furthermore if the facility is small, so that it can only hold a few vehicles at a time, requiring that vehicles be ‘swapped’ in and out for the various coats, then $H(T_S)$ will likely be a lot less than 1 where I is at a maximum.

The coordinated sharing algorithm will become even more complex, if only parts of the vehicle need to be brought in for a coat, such as doors, hoods, trunk lids, or roof sections, in a kind of operating−system equivalent of paging or segmentation. Further complexity will arise if we have different kinds of vehicle, some needing 2 coats, some 3 and some 4, and some for 1 minute some for 2 and some for 3, and some taking 8 minutes to dry, some 9 and some 10. Where there are different kinds of vehicle, it usually means that the line is being used to manufacture custom items.

This demonstrates what is well known in production engineering [7]. When production lines are used to manufacture custom items, throughput capacity can often be improved by clever sharing of production resources among the various threads of production. Often the coordinated resource−sharing procedure for the operation is extraordinarily complex, and must be managed by a computer system. As well, finding the resource−sharing algorithms can be a difficult task, often tackled by computer scientists.

It is an unfortunate fact that the sharing equation does not tell us what the resource−sharing algorithm will be. It tells us merely that we should look not only for such a procedure, but for a range of such procedures, in order of their ability to wring more and more throughput capacity out of the resource base as T_S increases. However, the equation does tell us how to go about selecting the optimum from among these procedures, and does warn us about the conditions that will cause the system to thrash, thrashing being a much more general phenomenon than might be expected. It will also tell us when it might be better to advise those responsible for the system throughput to find a way to increase R, and forego using a complex, coordinated, resource−sharing procedure.

It will always be the case that the simplest, although not necessarily the easiest, way to improve throughput capacity will be to increase R. The more complex way will always be to look for and use a coordinated resource−sharing procedure. However, since resources are always scarce, as each new type of system appears, it can be expected that research to find a way to increase the system throughput capacity by resource−sharing procedures will not be far behind.

5. Concluding Remarks

A basic system performance equation, called the resource sharing equation, relating throughput capacity, resources and a coordinated sharing procedure for system threads, and valid for all non−growth, non−feedback agent−directed systems, but especially for computer and information systems, have been presented and justified.

The resource sharing equation states (a) that throughput capacity increases linearly with valid increases in resources, and (b) that for a given throughput capacity, as system resources are decreased, system operating complexity must increase and vice versa. The equation can be used to infer Spreng's triangle.

In the equation the constants and variable parameters can be reduced to numbers and measurable quantities, so that the equation is subject to experimental verification. Nevertheless, the reader who has taken care to fully grasp the equation will no doubt see that it fits with his or her own experience of practical functioning systems. However, the major benefit of the equation to system designers is that it promotes and simplifies clear thinking and accurate reasoning about complex system situations and possibilities.

References

1. Anderson, T.E., Lazowska, E. D., and Levy, H. M., The performance implications of thread management alternatives for shared−memory multiprocessors, IEEE Trans. On Computers, 38(12), 1989, pp1631−1644.

2. Barrow, J.D., "Impossibility – The limits of science and the science of limits," Oxford University Press., Oxford, 1998, p 146.

3. Bradley, J. Five equations relating throughput capacity to system resources and risk for all agent−directed non−growth systems, Research Report No.: 99/642/05, Department of Computer Science, University of Calgary, 48 pages 1999 (available online).

4. Date C.J. An Introduction to Database Systems, 7th Edition, Addison−Wesley, Reading Mass., 2000.

5. Gray, J. and Reuter. A. Transaction processing, Concepts and Techniques, Morgan Kaufmann, Palo Alto, California, 1993.

6. Hennessy, J. L., and Patterson, D. A., "Computer Architecture: A Quantitative Approach", Morgan Kaufmann Publishers, Palo Alto, California, 1990

7. Hillier F.S. Introduction to Operations Research, McGraw−Hill, New York, 1995

8. Laowska, E.D, Zahorjan, J., Graham, G.S., and Sevcik, K. C., "Quantitative System Performance," Prentice−Hall, Englewood Cliffs, NJ, 1984.

9. Olsen, T. A., Ayhan H. Scheduling of multi−class single−server queues under non−traditional performance measures, Operations Research, Vol. 48 No 3 May−June 2000.

10. Sauer, C. H., and Chandy, K. M., "Computer System Performance Modeling", Prentice−Hall, Englewood−Cliffs, N.J., 1981.

11. Silberschatz, A, Peterson, J. L., "Operating Systems Concepts.", 5th Ed, Addison Wesley, Reading, Mass., 1998

12. Spreng, D.T. On time, information and energy conservation, ORAU/IEA−78−22(R), Inst. for Energy Analysis, Oak Ridge Assoc. Universities, Oak Ridge, Tennessee, 1978

13. Weinberg, A.M. On the relation between information and energy systems, National Conference of the ACM, 1980, reproduced in “Maxwell’s Demon”, H.S. Leff and A.F. Rex, editors, Princeton University Press, 1990, p. 116.

14. Zurek, W. Thermodynamic cost of computation, algorithmic complexity, and the information metric, Nature, Vol. 342, 1989, p. 119.

_