Journal of System Architecture

Vol. 43, n. 9, (1997)

pp. 605-624

**A Study of the Dynamic Behaviour of Some Workload Allocation Algorithms by Means of Catastrophe Theory[*]**

*F. A. Schreiber, M. Baiguera, G. Bortolotto, V. Caglioti*

*Dipartimento di Elettronica e Informazione - Politecnico di Milano*

**Abstract:** the dynamic behaviour of some workload allocation algorithms is studied to identify instability conditions induced either by an external overload situation or by a workload reallocation, triggered by the failure of some of the requested resources. We use concepts, drawn from that part of system theory which is known as Thom's Catastrophe theory, such as the Drift and the Potential Functions. The method is then applied to workload balancing in a Distributed Database sytem; however our method can be extended to other kind of systems which can be found in the telecommunication networks and in the distributed computing systems areas. We first deal with simple examples to show how the instability conditions show up, then we show its application to a well known allocation algorithm (Sender Initiated) which is known to be stable under overloads and this behaviour is confirmed.

*Keywords:* algorithms stability evaluation, Catastrophe theory, distributed Database, soft real-time, workload allocation and balancing

**Contact Author**

Prof. Fabio A. Schreiber
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Piazza Leonardo da Vinci 32
20133 MILANO
Italy

Phone: ++39-2-2399.3530
Fax: ++39-2-2399.3411
E-mail: schreibe@elet.polimi.it

[*] Work partially supported by CSISEI-CNR and MURST

## 1. Introduction

The behaviour of workload allocation and balancing algorithms influences, under particular conditions and system constraints, the dependability of distributed computing systems. In fact, in these systems usually a number of replicas of the resources exist, which can be considered equivalent from an application point of view. These replicated resources can be used either to balance the workload of the system under normal working conditions or to reallocate the workload to the *N-m* surviving resources in case of failure of *m* of them.

A particular case of such systems is constituted by a Distributed Database System (DDBS) with replicated data (the replicated resources) and with an upper threshold on the response time (the constraint) to the incoming transactions (the workload). This constraint makes the DDBS a *firm or a soft real-time system*, in the sense that the threshold in the response time should not be violated with a given (high) probability, but, in any case, no real harm results in aborting and restarting a limited number of delayed transactions or in accepting some tardy response

If the response time exceeds the limit at a node (an element containing processors and data), that component of the system is considered overloaded and its behaviour is considered faulty. In case of failure of one of the resources of a DDBS, a workload allocation algorithm automatically reassigns the workload among the surviving equivalent nodes. Recovery of a failed node is performed as soon as its workload is such as the response time threshold is respected (the physical failure of a node can be treated as a limit case).

The action of the load scheduler may, in some cases, lead to a sequence of failures, called "domino effect", which brings the system to a collapse situation. This feature of the behaviour of an algorithm is a form of system "instability". Other works addressed the problem of the stability of dynamic workload allocation and of scheduling algorithms from an algorithmic point of view [2, 3, 11, 15, 16]; however our purpose is *not to create a new algorithm, but to propose a general methodology for analysing the stability of existing algorithms.*

Therefore, in this work we investigate the possibility of using Catastrophe theory in the evaluation of the stability of load distribution algorithms in a DDBS; however our model is rather general and it can be extended to other kind of systems which can be found in the telecommunication networks and in the distributed computing systems areas [17]. Previous work by Nelson and Thomasian [9, 18] showed the applicability of Catastrophe theory to the dynamic behaviour of other kinds of algorithms; we feel that this could be a fruitful approach toward a general evaluation methodology of the correctness of algorithms under dynamic workload conditions, which blends dynamic systems theory and computer systems engineering. From the engineering point of view, *the methodology allows to evaluate either the maximum workload to which a given system can be safely exposed or, on the other hand, how large the set of the equivalent resources must be to safely support a given workload.*

Catastrophe theory [10] studies systems that, under particular conditions, show sudden changes (called *catastrophes*) in the steady equilibrium state as a consequence of small changes in the value of certain input parameters. We argue that algorithms' dynamic instability can be considered as a particular catastrophic behaviour; a small increase in the workload of a system can rapidly cause most of the processors to reach the failed state.

For the reader's convenience, we summarize in section 2 some of the basic results of Catastrophe teory, used in the paper. We then apply our methodology to the detailed study of a simple introductory algorithm, to show how the modelling is performed and how a poorly designed algorithm leads to a domino effect (section 3). In this example we suppose transactions to belong to a single class, requesting answers from a single resource, and the serving nodes to have identical service rates. Then we separately show how to extend the analysis to nodes with different service rates (section 4.1), to transactions belonging to different classes of service (section 4.2), and to transactions which require the availability of a sequence of nodes to be processed (section 4.3). As a last example, we apply the methodology to the analysis of the "Sender Initiated" workload balancing algorithm, which is known to be stable under overloading conditions [6], and this property is confirmed by the absence of catastrophic points (section 5). These examples are treated more concisely; an extended description can be found in [1].

The results obtained by analytical models, which always require some simplifying assumptions to be solved, have been validated by simulation runs and the maximum difference between the corresponding results has been within 10%.

## 2. Fundamentals of Catastrophe Teory

As it is well known from the elementary theory of dynamic systems, equilibrium states correspond to the maxima and minima of the system's *potential function*; maxima represent unstable equilibrium points while minima represent stable ones. A potential function $V(x,c)$ is a function of both the system state $x$ and of the control parameter(s) $c$. This function is defined such that the system, given a fixed $c$, makes a transition from a state $x$ to a state $x'$ if $V(x,c) \geq V(x',c)$. The *drift,* defined as $\frac{dx(t)}{dt}$, expresses the trend of the system state. As a consequence, a function $V(x,c)$ is a potential function if, for any fixed value of the control parameters $c$, the drift and the partial derivative $\frac{\partial V(x,c)}{\partial x}$ have opposite sign.

Therefore one can take $V(x,c)$ such that $\frac{dx(t)}{dt} = -\frac{\partial V(x,c)}{\partial x}$, i.e. $V(x,c) = -\int \frac{dx(t)}{dt} dx + K$

Given a fixed value of $c$, the system states for which $\frac{\partial V(x,c)}{\partial x}$ =0 are equilibrium states. An equilibrium state $\underline{x}$ is *stable* if, for a small deviation $x$-$\underline{x}$ from $\underline{x}$, the system state moves back to $\underline{x}$, i.e. $x$-$\underline{x}$ and the drift $\frac{dx(t)}{dt}$ have opposite sign and therefore $x$-$\underline{x}$ and $\frac{\partial V(x,c)}{\partial x}$ have the same sign; thus, $V$

*has a local minimum in* $\underline{x}$. Viceversa, if $V$ has a local maximum in $\underline{x}$, $\underline{x}$ is an *unstable* equilibrium point, since, for a small deviation of $x$ from $\underline{x}$, the state tends to further deviate from $\underline{x}$.

Catastrophe theory is based on the use of potential functions; a system has a catastrophic behaviour if the shape of its potential function is similar to the shape of one of *Thom's canonical formulas* (table 1).

| Canonical Formula | Name |
|---|---|
| $x^3 + ax$ | Fold |
| $x^4 + bx^2 + ax$ | Cusp |
| $x^5 + cx^3 + bx^2 + ax$ | Swallowtail |
| $x^6 + dx^4 + cx^3 + bx^2 + ax$ | Butterfly |

Table 1.

Two potential functions are defined to be similar if they have the same number, type and sequence of equilibrium states. While referring to [10] for a thorough presentation of Catastrophe theory, we give in the following only the basic principles we need for our work.

A catastrophe happens when a minimum and a maximum degenerate into an inflection point as a consequence of a small change in the value of one of the input parameters. As an example, let us consider the so-called *cusp* canonical potential, whose formula is:

$$V(x) = x^4 + bx^2 + ax$$

Drawing the graphs of $V(x)$ corresponding to increasing values of the parameter $a$, we can notice that the shape of the potential function and the steady equilibrium state change from two stable and one unstable equilibrium states to only one stable state, and that a catastrophe takes place (Figures 1.1-1.2-1.3-1.4). We also intuitively understand that, if we leave from the situation of figure 1.4 and decrease $a$ in order to simply revert to that of figure 1.1, the small circle, which represents the state of the system, will not move from its new position; it is necessary to go a symmetrical way, by further decreasing $a$, thus deepening the minimum in $x_1$ and rising that in $x_2$, to bring back the state of the system in $x_1$. This phenomenon accounts for the *hysteresis* of such kind of systems [8, 10].

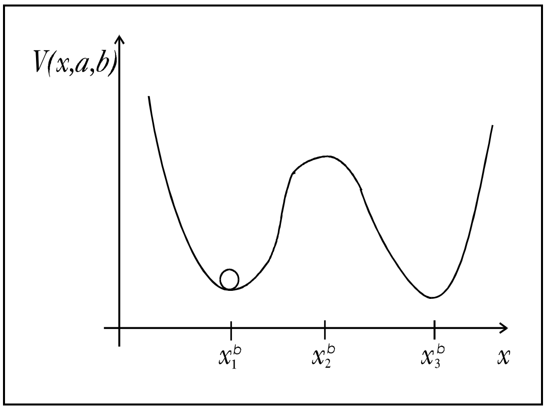


Figure 1.1

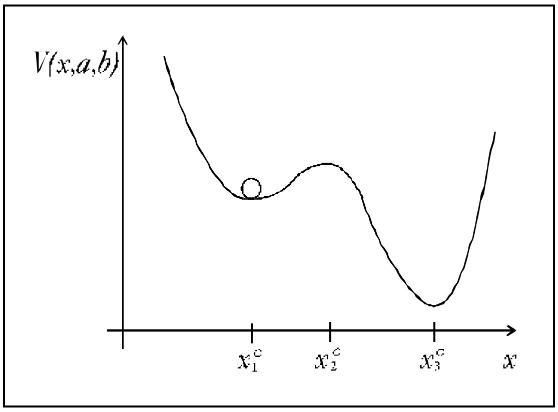


Figure 1.2

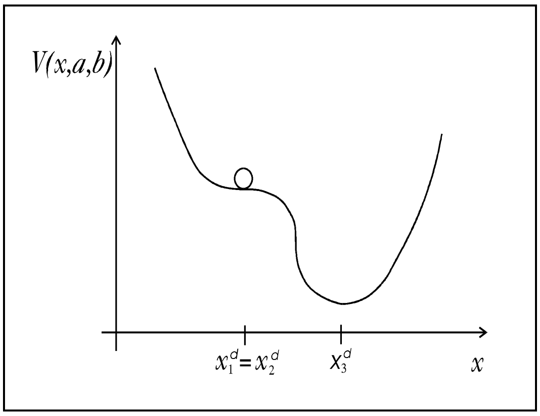


Figure 1.3

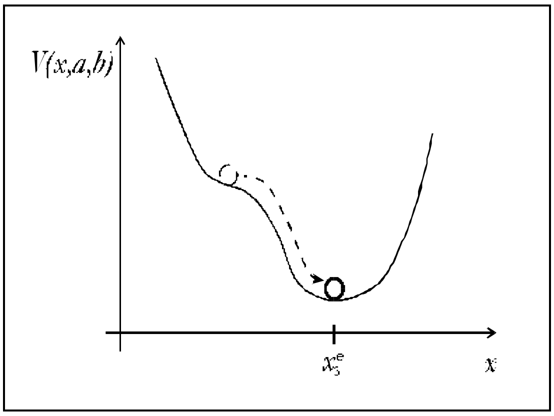


Figure 1.4

Therefore, if we want to make use of Catastrophe theory in studying the stability of load balancing algorithms, we need to obtain a potential function for the system. In order to do that, first we have to choose a state variable, *x(t)*, in such a way it is a quantity somehow related to the performances of the system we want to study.

We require that *x(t)* be a finite state Markov chain, which we approximate with a fluid approximation, i.e. we describe the stochastic process through its mean value [7]. In this work, we choose the *number of working resources* in the distributed database as the state variable *x(t)*. This is an easily observable variable and it is directly related to the "health" of the system. Then, let us define the *drift* of the system by a differential state equation of the form:

$$\frac{dx}{dt} = \alpha(x(t),c(t)) - \beta(x(t),c(t))$$

if $t$ is a continuous variable, or:

$$\frac{\Delta x}{\Delta t} = \frac{x(t+\Delta t)-x(t)}{\Delta t} = \alpha(x(t),c(t)) - \beta(x(t),c(t))$$

if $t$ is a discrete variable, and where $\alpha$ is the *birth rate* of *x(t)*, $\beta$ is the *death rate* of *x(t)*, and *c(t)* are the input parameters.

As we mentioned before, the solutions of the equation *drift(x(t),c(t))=0* are the equilibrium values of the state variable; they correspond to minima, maxima or inflection points of the *potential function*

$$V(x,c) = \int([\alpha(x(t),c(t)) - \beta(x(t),c(t))]).dx + C$$

A system has a catastrophic behaviour if the shape of its potential function is similar to the shape of one of the canonical formulas. Moreover, it is possible to point out the conditions (the values of the input parameters) at which the catastrophe (the instability) shows up.
After the analysis of the stability, we shall be able to fix a "safe" load for the distributed database.

### 3. Modelling a simple algorithm

We shall illustrate the analysis method based upon Catastrophe theory by studying the stability of an introductory workload distribution algorithm, named *ARF (Flow Reallocation Algorithm)*.
We consider a distributed database where transactions with soft real-time constraints, arriving from a common incoming stream, may be directed to one of *N* alternative nodes.
Each node of the distributed database is modelled as a single server and a queue. If a node becomes overloaded, the existing jobs remain in its queue, while if a node is subjected to a hardware failure, the jobs are transferred to one of the working nodes: so no transaction is lost. After being processed at a server, transactions leave the system.
We use queuing theoretic results in analysing the stability of the algorithm. In particular, we suppose that processors have exponentially distributed inter-arrival and service times (M/M/1 queue). These hypotheses, which are very common to allow for solvability of analytical models, cannot always be a faithful representation of the real world. Therefore we compared the results of the analytical model with those obtained by simulating the system with the QNAP2 simulation package, as we mention in the following.
We can translate the soft real-time constraint into a constraint on the queue length at each node [5, 6, 12, 14, 19], requiring it to be kept below a certain threshold length. We define:

$$\tilde{q} = t_{resp} \cdot \mu$$

where $\tilde{q}$ is the maximum allowed instantaneous queue length, $t_{resp}$ is the maximum allowed response time for a transaction and $\mu$ is the mean service rate. An element of the distributed database is considered to be overloaded (failed) if $\tilde{q}$ or more transactions are simultaneously waiting to be processed at it. The behaviour of *ARF* is the following: periodically (every *T* seconds) the algorithm checks if working nodes have become overloaded ($q \geq \tilde{q}$), and if nodes which were previously in the failed state have recovered. *ARF* distributes workload only among the working nodes, in such a way all of them have the same overloading (failure) probability.
The drift equation is based on the probability that $q \geq \tilde{q}$. The Queueing theory formulas used are only valid if the transient phenomena due to a change in the mean arrival rate of transactions, are

exhausted; this requires for a large value of $T$. However, a too large $T$ prevents ARF from detecting temporary overloads: therefore a presimulation is needed in order to select an appropriate value for $T$. This presimulation has been performed, and an appropriate $T$ has been selected by evaluating how accurately the drift equation predicts the equilibrium states [1].

```
program  ARF  ;
let      N the set of nodes;
         X(t+1) set of working nodes at time t+1;
         X(t) set of working nodes at time t;
         μ mean service rate to the single node;
         λ mean arrival rate to the system;
         q̃  maximum number of clients in a node;
begin
X(t+1):=X(t);
for each time interval T do
         for each node j of X(t+1) do
                  if  clients in node j  ≥  q̃ ;
                           then   j  :=  timed-out;
                                    X(t+1):= (X(t+1) - node j);
                  end (*if*)
         end (*for each*)
         for each node j of (N-X(t) ) do
                  if  clients in node j  >  0
                           then  empty all clients in node j
                                    j:= working;
                                    X(t+1)   =  X(t+1)  +   node j  ;
                  end (*if*)
         end (*for each*)
end (*for each*)
         each arriving transaction do
                  destination:= random (1..X);
                  send transaction to destination;
         end (*each*)
end (*program ARF*).
```

Such an allocation policy can be obtained by sending the transactions randomly to the working nodes, if we suppose the distributed system to be made up of $N$ identical processors and all of the transactions to request the same service time (afterwards, we will analyse the stability of *ARF* in the case it allocates workload in a distributed system made up of elements having different service rates, and in the case *ARF* distributes transactions requiring different service times). A failed element is considered to be recovered when its queue length returns below the threshold.

In order to obtain very simple equations for studying the stability of the algorithm, we suppose that every component found to be overloaded at a check becomes working again within the next check. This can be obtained restarting transactions exceeding the threshold $\tilde{q}$ on working nodes.

By the way, we can notice that this model is very similar to that of a toll-barrier at the end of a highway (figure 2).

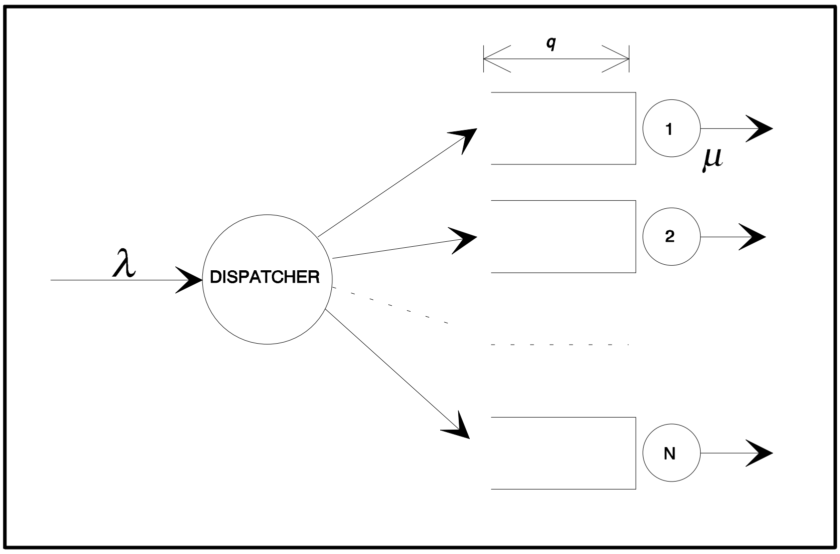


Figure 2

As we mentioned above, we consider the number of working resources at time *t* to be the state variable *x(t)*. It can be shown that *x(t)* is a discrete-time Markov chain.

Let us define:

- *N*: the number of nodes in the distributed database
- $\mu$: the mean service rate of a node
- $\lambda$: the mean arrival rate of transactions to the database
- $\tilde{q}$: the maximum allowed number of transactions to be served at a node (process queue length)
- *T*: the time interval between two checks
- *x(t)*: the number of working nodes

(we shall simply indicate *x(t)* as *x* in the equations that follow. In order to have a "lighter" notation, we also drop the explicit indication of the dependence of the potential and of the birth and death rates from the input parameters *c*. These parameters are kept constant in each experiment, while *x* is the current independent variable).

In order to get the drift and the potential function, we have to obtain the expressions of the mean birth rate and of the mean death rate for the system.

The *birth rate* $\alpha(x)$ at time *t* corresponds to the average number of failed elements that become working again in a second. According to the hypothesis, α is given by:

$\alpha(x) = (N - x)/T$

where *N-x* is the number of overloaded nodes in the distributed database at time *t*.

The *death rate* $\beta(x)$ at time $t$ corresponds to the average number of working nodes that become overloaded in a second.

According to the theory of M/M/1 queues, the probability that $\tilde{q}$ or more processes are present (and not yet completed) at a service centre is:

$$P[q \geq \tilde{q}] = \rho^{\tilde{q}} \qquad \text{if } \rho < 1$$

$$P[q \geq \tilde{q}] = 1 \qquad \text{if } \rho \geq 1$$

where $\rho$ is the utilization factor of a node (utilization factor = mean arrival rate / mean service rate).

We assume the elements of the distributed database to have the same mean service rate, and we know that *ARF* distributes the incoming stream of $\lambda$ jobs/second among the $x$ working servers choosing randomly the destinations of the transactions; so we have:

$$P[q \geq \tilde{q}] = \left(\frac{\lambda/x}{\mu}\right)^{\tilde{q}} = \left(\frac{\lambda}{\mu \cdot x}\right)^{\tilde{q}} \qquad \text{if } \rho < 1$$

The number of working elements found to be overloaded at a check is simply:

$$x \cdot P[q \geq \tilde{q}] = x \cdot \left(\frac{\lambda}{\mu \cdot x}\right)^{\tilde{q}} \qquad \text{if } \rho < 1$$

Then, the average number of working nodes found to be failed in a second, i.e. the death rate $\beta$, is given by:

$$\beta(x) = \frac{x}{T} \cdot \rho^{\tilde{q}} = \frac{x}{T} \cdot \left(\frac{\lambda}{\mu \cdot x}\right)^{\tilde{q}} \qquad \text{if } \rho < 1$$

$$\beta(x) = \frac{x}{T} \cdot 1 \qquad \text{if } \rho \geq 1$$

The drift of the system is then:

$$drift(x) = \alpha(x) - \beta(x) = \frac{N - x}{T} - \frac{x}{T} \cdot \left(\frac{\lambda}{\mu \cdot x}\right)^{\tilde{q}} \qquad \text{if } \frac{\lambda}{\mu} < x \leq N$$

$$drift(x) = \alpha(x) - \beta(x) = \frac{N - x}{T} - \frac{x}{T} = \frac{N - 2x}{T} \qquad \text{if } 0 \leq x \leq \frac{\lambda}{\mu}$$

In figure 3 we show the drift graph under the following conditions:

$$\begin{cases} N = 60 \text{ nodes} \\ \mu = 2\ job/\sec \\ \lambda = 74\ job/\sec \\ \tilde{q} = 7\ job \end{cases}$$

In the graph we can notice the presence of three zero points, this is typical for a cusp catastrophe. In fact $x_1$ and $x_3$ correspond to minima points and $x_2$ to a maximum.

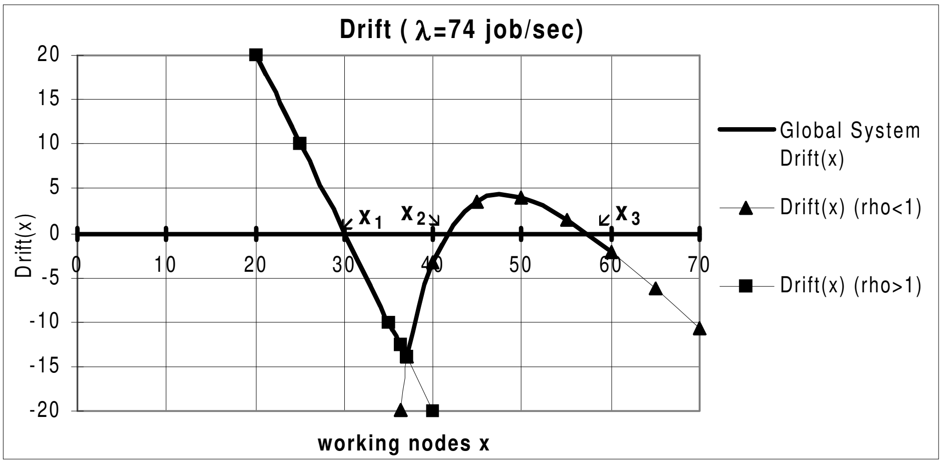


Figure 3

The potential function for the system is the following:

$$V(x) = -\frac{1}{T}\left\{N\cdot x - \tfrac{1}{2}x^2 - \left(\frac{\lambda}{\mu}\right)^{\tilde{q}}\cdot\left(\frac{x^{2-\tilde{q}}}{2-\tilde{q}}\right)\right\} + C_1\ , \qquad \text{if } 0 \le \left(\frac{\lambda}{\mu\cdot x}\right) < 1$$

$$V(x) = -\frac{1}{T}\left\{N\cdot x - x^2\right\} + C_2\ , \qquad \text{if } \left(\frac{\lambda}{\mu\cdot x}\right) \ge 1$$

To assure the continuity of *V(x)* in $\left(\lambda/\mu\cdot x\right) = 1$, the equality:

$$-\frac{1}{T}\left\{N\cdot x - \tfrac{1}{2}x^2 - \left(\frac{\lambda}{\mu}\right)^{\tilde{q}}\cdot\left(\frac{x^{2-\tilde{q}}}{2-\tilde{q}}\right)\right\} + C_1 = -\frac{1}{T}\left\{N\cdot x - x^2\right\} + C_2$$

must be satisfied, so determining the value of $C_1 + C_2$.

The above two expressions for *V(x)*, namely one for ρ< 1 and one for ρ ≥ 1, are drawn (thin lines) in Fig. 4.1. Starting from these expressions, the potential function (bold line) is built: for $x>\lambda/\mu$ the ρ< 1 expression is valid, while for $x \le \lambda/\mu$ the ρ≥ 1 expression holds.

According to Catastrophe theory, we can foresee whether (and when) a catastrophe happens in the system, i.e. if and in which conditions of workload the algorithm shows its unstable behaviour. This can be done by studying the shape of the potential function.

The maximum degree of the *V(x)* polynomial depends on $\tilde{q}$; therefore it could be hard to recognize in it one of the canonical formulas of table 1. However, the similitude of *V(x)* with the canonical formulas is required only "in the small" in the range where the zeroes of the drift function are located. We can see, by plotting *V(x)* in this region that it has a quartic behaviour with control parameters $\lambda$ and $\tilde{q}$, similar to that of the cusp catastrophe.

More precisely, we are interested in the number and type (steady or not) of equilibrium states, as a function of the value of the "input parameter" $\lambda$.

The graphs of potential function are obtained using constant values for *N*, $\mu$, $\tilde{q}$, and incrementing $\lambda$. We consider $\lambda$ as the input parameter (independent variable) because it is partially uncontrollable, so it can grow with a relatively high rate, and more dynamically with respect to *N*, $\mu$ and $\tilde{q}$.

Drawing the graphics of *V(x(t)),* for a system of $N$=60 nodes with a $\mu$=2 job/sec, and $\tilde{q}=7$ jobs, associated to increasing values of the workload $\lambda$, we can observe that the potential function has a varying number of equilibrium points (Figures 4.1, 4.2, 4.3, 4.4, 4.5): a minimum ($x_3$), a minimum ($x_3$) and an inflection point, two minima ($x_1$ and $x_3$) and a maximum ($x_2$), a minimum ($x_1$) and an inflection point, a minimum ($x_1$).

We can notice that, in correspondence with $\lambda=\lambda^*$, the minimum $x_3$ merges with the maximum $x_2$ to give rise to an inflection point, and that the only available steady state for the system is the minimum $x_1$.

If the workload is lower than $\lambda^*$, the mean number of working nodes in the distributed database is $x_3$; if the workload is higher than $\lambda^*$, the mean number of non-overloaded nodes is $x_1$.

When $\lambda=\lambda^*$, a sequence of failures ("domino effect") makes the equilibrium state of the system suddenly change from $x_3$ to $x_1$: the algorithm shows an unstable behaviour, precisely in correspondence of the value $\lambda^*$ of the transactions' arrival rate to the system.

Moreover, we have to observe that the system has unacceptable performances when its steady equilibrium state value is $x_1$. In fact, since $x_1=N/2$, it is:

$N-x = x= N/2$

It means that all the nodes which were operative at time *t* are found to be overloaded at the test at time $t+T$.

The soft real-time constraint is never satisfied, and the response time for the transactions is always above the threshold: the system has reached a collapse situation.

Finally, we can set the "safe load" for the system as , for example, the 80% of $\lambda^*$.

Comparison of the analytic results with simulation runs for the ARF algorithm shows they are conservative and that the error is within the 5% limit. For instance, the simulation results shown in figure 5 were obtained with the following values for the parameters, where *D* is the duration time of each simulation run: $N$=60 nodes; $\mu$=2 job/sec; $\tilde{q}=7$ jobs; $D$=2000 sec; $T$=20 sec.

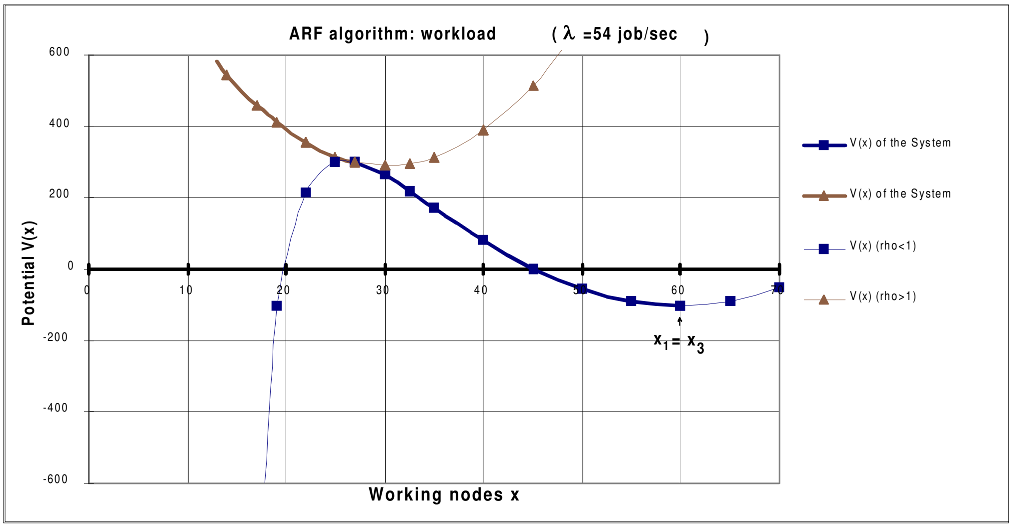


Figure 4.1

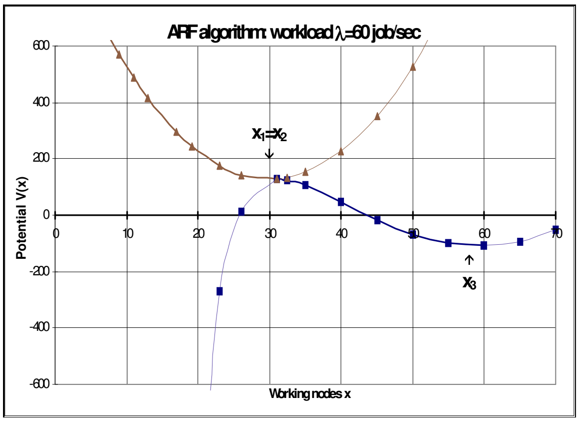


Figure 4.2

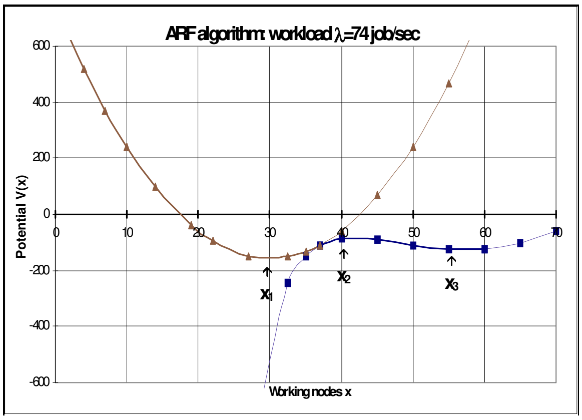


Figure 4.3

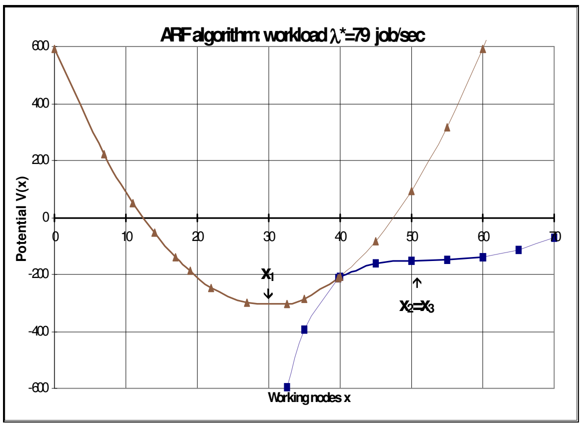


Figure 4.4

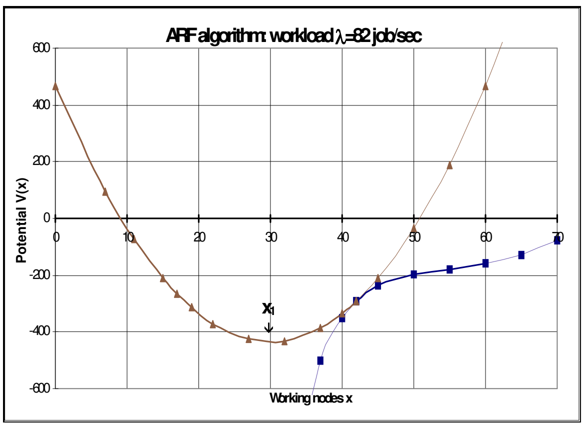


Figure 4.5

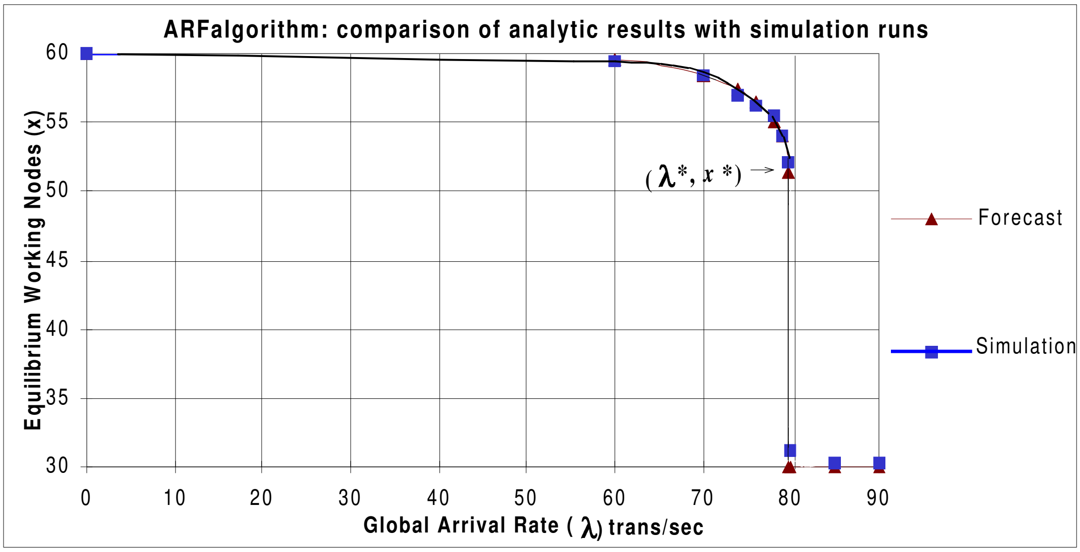


Figure 5

## 4 Extensions to the ARF algorithm

In the next sections we consider the following extensions to the ARF algorithm:

- DDB made up of nodes having different mean service rates
- transactions requiring different service times
- transactions requiring the existence of “paths” (sequences of nodes) to be processed

In each case we can define a birth rate and a death rate, and therefore we can write a drift equation and, from it, derive a potential equation.

Several simplifying assumptions are discussed, which are required for the analytical solvability of the problems.

### 4.1 Extension of ARF algorithm to DDB made up of nodes having different mean service rates

In this section we consider a DDB made up of nodes having different mean service rate. We still use ARF algorithm to balance the load arriving to the DDB. This case also demonstrates that ARF is characterised by a “catastrophic behaviour”.

Let us assume the following hypotheses:

- N nodes with different mean service rates $\mu_j=\mu(n_j)$;
- alternative nodes;
- one class of transactions;
- input workload to the DDB equal to $\lambda$ job/sec.

The total load is distributed by the algorithm in such a way as the overload probability $C$ is the same for all the nodes. This is obtained imposing the following condition on the response time $t_{resp}$ :

$$P_i[\,t_{resp} \ge \tilde{t}\,] = P_j[\,t_{resp} \ge \tilde{t}\,]\text{, with } i \ne j \text{ and i, j= 1 ... } x$$

We define $\lambda_i$ as the value of the load that ARF sends to the i-th node.

Basing on queuing theory [KLE76], we can use the following approximation of the overload probability $P_i[t_{risp} \ge \tilde{t}\,]$:

$$P_i[\,t_{resp} \ge \tilde{t}\,] = e^{-(1-\rho_i)\mu_i \cdot \tilde{t}} \qquad \text{if } \rho < 1$$

$$P_i[\,t_{resp} \ge \tilde{t}\,] = 1 \qquad \text{if } \rho \ge 1$$

where $\rho_i = \left(\dfrac{\lambda_i}{\mu_i}\right)$. The approximation is satisfactory only if the load value $\lambda$ is sufficiently high, but this is the very situation we are interested to.

We assume that there are no transactions which are not allocated to any nodes, and this is expressed by the following relation:

$$\sum_1^x {}_i\lambda_i = \lambda$$

Now, basing on the preceding relations, we can write the following systems:

$$\begin{cases} e^{-(1-\rho_i)\mu_i \cdot \tilde{t}} = C \\ \sum_1^x {}_i\lambda_i = \lambda \end{cases} \qquad \text{if } \rho < 1$$

$$\begin{cases} 1 = C \\ \sum_1^x {}_i\lambda_i = \lambda \end{cases} \qquad \text{if } \rho \ge 1$$

Resolving the system, for $\rho < 1$, with respect to $\lambda_i$ and C'(where $C' = -\ln(C)$) we obtain .

$$\begin{cases} \lambda_i = \left(\mu_i - \dfrac{C'}{\tilde{t}}\right) \\ C' = \dfrac{\sum_1^x {}_i\mu_i}{x} \cdot \tilde{t} - \dfrac{\lambda}{x} \cdot \tilde{t} \end{cases}$$

We notice that for $\rho \ge 1$ the system is verified $\forall \lambda_i$.

Each node has the same overload probability, then we cannot exactly know which of them will become overloaded. Moreover in the system of equations we use the following approximate relation:

$$\overline{\mu} = \frac{\sum_1^N {}_i\mu_i}{N} \cong \frac{\sum_1^x {}_i\mu_i}{x}$$

It tells that the average value of the mean service rate is about constant. This approximation is effective when $x$ and $N$ are sufficiently high.

The solution of the system is the following:

$$\begin{cases} \lambda_i = \left(\frac{\lambda}{x} + \mu_i - \overline{\mu}\right) \\ C = e^{-\left(1-\frac{\lambda}{\overline{\mu}\cdot x}\right)\overline{\mu}\cdot\tilde{t}} = e^{-(1-\overline{\rho})\overline{\mu}\cdot\tilde{t}} \end{cases}$$

where $\overline{\rho} = \frac{\lambda}{\overline{\mu}\cdot x}$.

We notice that the approximation is valid only when the following relation is true:

$$\lambda_i = \left(\frac{\lambda}{x} + \mu_i - \overline{\mu}\right) \geq 0 \qquad \text{for } i = 1 \dots x$$

As we noticed before, this is true if the workload is high.

*Birth Rate* $\alpha(x)$

The birth rate has not changed from what we have seen in the preceding examples:

$$\alpha(x) = \frac{N-x}{T}$$

*Death Rate* $\beta(x)$

The mean number of servers which become not operative corresponds to the sum of the overload probabilities of each node.

$$\textit{mean number of nodes found overloaded at a control} = \sum_{i=1}^{x} P_i[\, q_i \geq \tilde{q}_i \,] = \sum_{i=1}^{x} C = x\cdot C = x\cdot e^{-(1-\overline{\rho})\overline{\mu}\cdot\tilde{t}}$$

Therefore, the expression of the death rate results the following:

$$\beta(x) = \frac{x}{T}\cdot C = \frac{x}{T}\cdot e^{-(1-\overline{\rho})\overline{\mu}\cdot\tilde{t}} \qquad \textit{if} \quad \overline{\rho} < 1$$

$$\beta(x) = \frac{x}{T}\cdot 1 \qquad \textit{if} \quad \overline{\rho} < 1$$

*Drift*

$$\textit{drift} = \alpha(x) - \beta(x) = \frac{N-x}{T} - \frac{x}{T}\left(\frac{\lambda}{\overline{\mu}\cdot x}\right)^{\overline{\tilde{q}}} = \frac{N-x}{T} - \frac{x}{T}(\overline{\rho})^{\overline{\tilde{q}}} \qquad \text{if } 0 \leq \left(\frac{\lambda}{\overline{\mu}\cdot x}\right) < 1$$

$$\textit{drift} = \alpha(x) - \beta(x) = \frac{N-x}{T} - \frac{x}{T}\cdot 1 \qquad \text{if } \left(\frac{\lambda}{\overline{\mu}\cdot x}\right) \geq 1$$

where $\overline{\tilde{q}}$ is the mean value of the threshold length $\tilde{q}_i$: $\overline{\tilde{q}} = \frac{\sum_{i=1}^{x}\tilde{q}_i}{x}$

*Potential*

The analytical expression of the Potential Function has a form which is already known:

$$V(x) = -\frac{1}{T}\cdot\left\{N\cdot x - \tfrac{1}{2}x^2 - \left(\frac{\lambda}{\overline{\mu}}\right)^{\overline{\tilde{q}}}\cdot\left(\frac{x^{2-\overline{\tilde{q}}}}{2-\overline{\tilde{q}}}\right)\right\} + C_1 \qquad \text{if } 0 \le \left(\frac{\lambda}{\overline{\mu}\cdot x}\right) < 1$$

$$V(x) = -\frac{1}{T}\cdot\left\{N\cdot x - x^2\right\} + C_2 \qquad \text{if } \left(\frac{\lambda}{\overline{\mu}\cdot x}\right) \ge 1$$

To assure the continuity of $V(x)$ in $\left(\frac{\lambda}{\overline{\mu}\cdot x}\right)$ =1, the sum $C_1$+$C_2$ must satisfy

$$-\frac{1}{T}\cdot\left\{N\cdot x - \tfrac{1}{2}x^2 - \left(\frac{\lambda}{\overline{\mu}}\right)^{\overline{\tilde{q}}}\cdot\left(\frac{x^{2-\overline{\tilde{q}}}}{2-\overline{\tilde{q}}}\right)\right\} + C_1 = -\frac{1}{T}\left\{N\cdot x - x^2\right\} + C_2$$

This Potential Function shows a quartical form in the interval $0\le x\le N$, so the system presents the same type of catastrophic behaviour analysed in precedence [1].

**4.2 Extension of the ARF algorithm to transactions requiring different service time (classes of transactions)**

In this section we extend the study of the stability of ARF algorithm to DDB serving different classes of transactions: every transaction member of a class has the same mean service time . We use the same approach seen in the preceding examples.

Let us suppose the following hypotheses:

- $N$ identical and alternative nodes;
- $J$ classes of transactions;
- Mean arrival rate to the DDB: $\lambda$ transactions per second;
- Mean arrival rate to the DDB of class $j$ equal to $\lambda_j$ transactions per second, where: $\lambda = \sum_{j=1}^{J} \lambda_j$ ;

ARF algorithm allocates transactions to the operative nodes $x$ using a random policy. This ensure an identical mean arrival rate to each node and therefore an identical overload probability.

Moreover we define:

- mean arrival rate to each node = $\lambda/x$ (job/sec)
- mean arrival rate of class $j$ to each node = $\lambda_j/x$ (job/sec)
- utilization factor of a node for class $j$ of transactions $\rho_j = \frac{\lambda_j}{\mu_j\cdot x}$
- utilization factor of a node $\rho = \sum_{j=1}^{J}\rho_j = \sum_{j=1}^{J}\frac{\lambda_j}{\mu_j\cdot x}$

In a system having only one class of transactions, the overload probability of a node is expressed by equation $P[q \ge \tilde{q}] = \rho^{\tilde{q}}$. Since in this case we have $J$ classes of transactions, We can not use this relation. Moreover it is not analytically trivial to obtain the exact overload probability of the system, so we choose to approximate that probability. The approximation used is an opportune weighted mean based on the simple mathematical form of the relation $\rho^{\tilde{q}}$.

So we use the following approximated overload probability:

$$P[t_{risp} \geq \tilde{t}\,] \cong \sum_{j=1}^{J} (h_j \cdot \rho^{\tilde{q}_j}) \qquad \text{if } \rho < 1$$

$$P[t_{risp} \geq \tilde{t}\,] = 1 \qquad \text{if } \rho \geq 1$$

where $h_j = \dfrac{\rho_j}{\rho}$.

This parameter represents the ratio of the utilization factor of the class $j$ of transactions to the total utilization factor. It is reasonable to suppose that a high value of $h_j$ entails a high value of overload probability.

*Birth Rate* $\alpha(x)$

The birth rate maintains the same form seen in the preceding examples:

$$\alpha(x) = \frac{N - x}{T}$$

*Death Rate* $\beta(x)$

$$\beta(x) = \frac{x}{T} \cdot \sum_{j=1}^{J} (h_j \cdot \rho^{\tilde{q}_j}) \qquad \text{if } \rho < 1$$

$$\beta(x) = \frac{x}{T} \cdot 1 \qquad \text{if } \rho \geq 1$$

*Drift*

$$drift = \alpha(x) - \beta(x) = \frac{N - x}{T} - \frac{x}{T} \cdot \sum_{j=1}^{J} \left( \frac{\rho_j}{\rho} \cdot \rho^{\tilde{q}_j} \right) \qquad \text{if } \rho < 1$$

$$drift = \alpha(x) - \beta(x) = \frac{N - x}{T} - \frac{x}{T} \cdot 1 \qquad \text{if } \rho \geq 1$$

*Potential*

The analytical expression of the Potential Function shows a form which is similar to the ones we have already seen in the preceding cases:

$$V(x) = -\frac{1}{T} \cdot \left\{ N \cdot x - \tfrac{1}{2} x^2 - \left( \frac{l}{\overline{m}} \right)^{\bar{\tilde{q}}} \cdot \left( \frac{x^{2-\bar{\tilde{q}}}}{2 - \bar{\tilde{q}}} \right) \right\} + C_1 \qquad \text{if } \rho < 1$$

$$V(x) = -\frac{1}{T} \cdot \left\{ N \cdot x - x^2 \right\} + C_2 \qquad \text{if } \rho \geq 1$$

With similar arguments as those used in the preceding sections, we can conclude that the Potential function maintains a quartical shape, that is it is similar to the Thom's canonical "cusp" shape, therefore the system presents the same type of catastrophic behaviour analyzed in precedence [1].

**4.3 Extension of ARF algorithm to transactions requiring the existence of "paths" to be processed**

In the preceding examples We used a simplified model of DDB constituted of equivalent nodes. In a real DDB nodes are connected by means of a computer network and each arriving transaction can be processed by a definite sequence of nodes, named "path". We exactly define a path as an ordered list of nodes. The study of a complete model of a computer network is hard and it is not in our aim, therefore

let us assume the following hypotheses in order to reduce the complexity of the model:

- *P* identical alternative paths;
- each path is made up of *l* identical nodes;
- one class of transactions;
- DDB subjected to a load of $\lambda$ transactions/sec.

ARF algorithm allocates load to various paths using a random policy. We consider a path to be not working if at least one of its nodes is overloaded.

*Birth Rate* $\alpha(x)$

The birth rate corresponds to the number of paths which return to be working per time unit:

$$\alpha(x) = \frac{(P - x)}{T}$$

where *x* is now the number of working paths .

*Death Rate* $\beta(x)$

The death rate corresponds to the number of paths that, on average, becomes non operational per time unit. Since paths are identical, we can obtain such a rate multiplying *x* and the overload probability for a path. Basing on a simple rule of the probability calculus We obtain the following expression:

$$\beta(x) = \frac{x}{T}\left(1 - (1 - \rho^{\tilde{q}})^l\right) \qquad \text{if } \rho < 1$$

$$\beta(x) = \frac{x}{T} \cdot 1 \qquad \text{if } \rho \geq 1$$

where $\rho = \frac{\lambda}{\mu \cdot x}$

*Drift*

$$drift = \alpha(x) - \beta(x) = \frac{P-x}{T} - \frac{x}{T}\left(1-(1-\rho^{\tilde{q}})^{l}\right) \qquad \text{if} \qquad 0 \le \rho < 1$$

$$drift = \alpha(x) - \beta(x) = \frac{P-x}{T} - \frac{x}{T} \cdot 1 \qquad \text{if} \qquad \rho \ge 1$$

```
program  ARFP

let  N          set of nodes;
     P          set of paths;
     X(t+1)     set of working paths at time t+1;
     X(t)       set of working paths at time t;
     l          path length;
     μ          service rate of a node;
     λ          system arrival rate;
     q̃          maximum number of clients in a node;
begin
X(t+1):=X(t);
for each interval of time T do
      for each  node of X(t)  do
          if  queue node n_ij  >  q̃_ij
             then  node n_ij := time-out;
                path j  := not working;
                X(t+1) :=  X(t+1)  -  path j;
          end (*if*)
      end (*for each*)

      for each  node n_ij "time-out" of (P - X(t))  do
          if  transactions in node n_ij
             then        empty  transactions in nodes  n_ij  ;
          end (*if*)
          if each node of the path j is working
             then        path j := working;
                         X(t+1) :=  X(t+1) + path  j;
          end (*if*)
      end (*for each*)

        each arriving transaction do
                 destination:=random(1..X);
                 send transaction to destination;
        end (*for each*)

end (*for each*)

end(*program ARFP*).
```

*Potential*

Obtaining the analytical expression of the Potential Function is not trivial, so we draw it using graphical software tools, and we discover that it has a quartical shape, typical of the "cusp" catastrophe. Therefore, the system presents the same type of catastrophic behaviour analyzed in the preceding examples [1].

```
program Sender-Initiated
let  N set of nodes
        n_i i-th node
        s_i number of clients in node n_i
        s̃  maximum number of clients in a node
begin
        for each node n_i do
            receive value s_i from local processor n_i
            if s_i ≥ s̃
                then
                    for each node n_j  (j≠i) do
                        get the value s_j from remote processor n_j,
                    end (*for each*)
                    choose the less loaded processor n_j*;
                    send transaction to processor n_j*;
                    receive results of transaction from n_j*;
                else execute transaction locally on processor n_i
            end(*if*)
        end(*for each*)
end(*program Sender-Initiated*).
```

## 5 Analysis of the "Sender-Initiated" Algorithm

At this point we are going to apply our methodology to a well known dynamic load balancing algorithm. We chose the "Sender-Initiated" algorithm because of its good performance and because it is stable up to saturation [4, 6].

The algorithm sends exceeding transactions from an overloaded node to nodes that respect soft real-time constraints. Let us define $\tilde{s}$ as the maximum allowed number of transactions to be served at a node (soft real-time constraint).

It is quite simple to explain how "Sender-Initiated" works: transactions arrive to nodes, then if the number of users in the service centre is greater than $\tilde{s}$, the exceeding transactions are sent to the processor with the lowest queue length. The control of queue length may be implemented using centralized or distributed methods, but this is not important for our stability analysis.

Let us suppose the following hypotheses:

- $N$ identical alternative nodes;
- one class of transactions;
- input workload to the DDB equal to $\lambda$ job/sec.

Let the state variable $x$ be the number of nodes that are not overloaded. It can be shown that $x$ describes a finite state Markov chain. We approximate $x$ by means of a fluid approximation.

*Birth Rate $\alpha(x)$*

Nodes remain overloaded on average for $1/\mu$ seconds. This is the time needed to process a transaction or rather it is the time to reduce queue length just under the soft real-time threshold $\tilde{s}$. Thus the number of nodes that becomes operational per time unit is:

$$\alpha(x) = \frac{N-x}{1/\mu} = (N-x)\cdot\mu$$

*Death Rate $\beta(x)$*

A node can not keep, waiting in queue or being served, more than $\tilde{s}$ transactions, thus we can model a service centre using an M/M/1/K queue, where K=$\tilde{s}$. Death rate can be obtained multiplying arrival rate $\lambda$ and the following probability:

$$p_{\tilde{s}-1} = \frac{(1-\rho)\rho^{\tilde{s}-1}}{1-\rho^{\tilde{s}}},$$

where $\rho = \frac{\lambda}{\mu \cdot x}$. This is the probability that exactly $\tilde{s}-1$ transactions can be found in a node.

The arrival rate $\lambda$ can be considered the maximum death rate. Therefore $\lambda \cdot p_{\tilde{s}-1}$ corresponds to the average number of nodes that become not working per time unit.

$$\beta(x) = \lambda \cdot \frac{(1-\rho)\rho^{\tilde{s}-1}}{1-\rho^{\tilde{s}}} \qquad \text{if } x > 0$$

$$\beta(x) = \lambda \cdot 1 = \lambda \qquad \text{if } x = 0$$

*Drift*

The drift expression is the following:

$$drift = (N-x)\cdot\mu - \lambda \cdot \frac{(1-\rho)\rho^{\tilde{s}-1}}{1-\rho^{\tilde{s}}} \qquad \text{if } x > 0$$

$$drift = (N-0)\cdot\mu - \lambda = N\cdot\mu - \lambda \qquad \text{if } x = 0$$

In this case the load balancing algorithm has no catastrophic behaviour, in fact with $\lambda < N\cdot\mu$ $(x > 0)$ the solution of *drift(x)=0* always has only one minimum, so the system has a unique steady state and no unstable state. When the arrival rate $\lambda$ is increased from 0 to $N\cdot\mu$, the steady state $x$ moves from the value of $N$ to $0$ without "jumping", i.e. the presence of only one zero in the drift equation implies that the system equilibrium state changes without sudden variation; we remember that a catastrophic system must have at least three real zeroes in the drift equation.

When $\lambda \geq N\cdot\mu$, no equilibrium state can exists because the arrival rate $\lambda$ exceeds the service rate of the whole DDB ($N\cdot\mu$). In other words, any load balancing algorithm can not satisfy the soft real-time constrain when the transaction inputs exceed the maximum processing capability of the system because the queue length of the nodes can not be limited, so no working equilibrium state is possible.

Therefore the study of the drift of the Sender-Initiated algorithm shows that the system is not characterised by catastrophic behaviour in the variable state $x$.

*Potential*

Obtaining a general analytical form of the Potential function is not trivial. In fact there are some mathematical difficulties in integrating the drift formula. This problem has been solved using graphical tools to draw the Potential Function (figures 6.1, 6.2, 6.3, 6.4, 6.5, 6.6).

The Potential Function shows a parabolic shape, therefore the system has no catastrophic behaviour, and the algorithm is stable up to the saturation workload for the whole DDB.

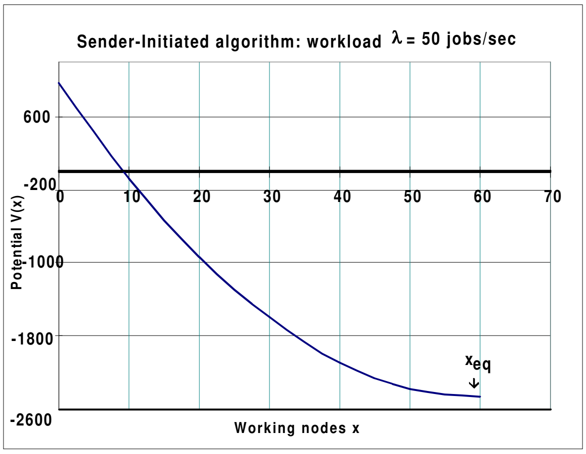


Figure 6.1

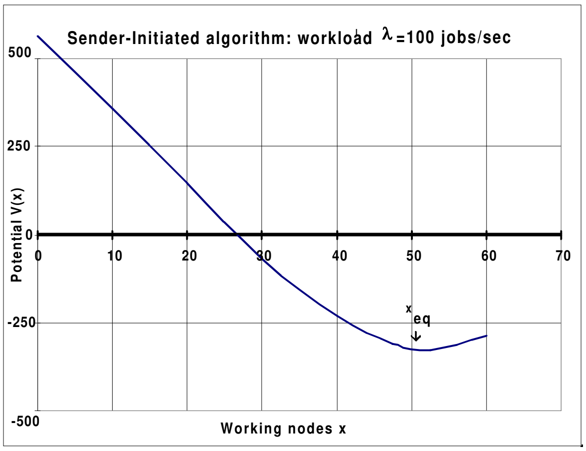


Figure 6.2

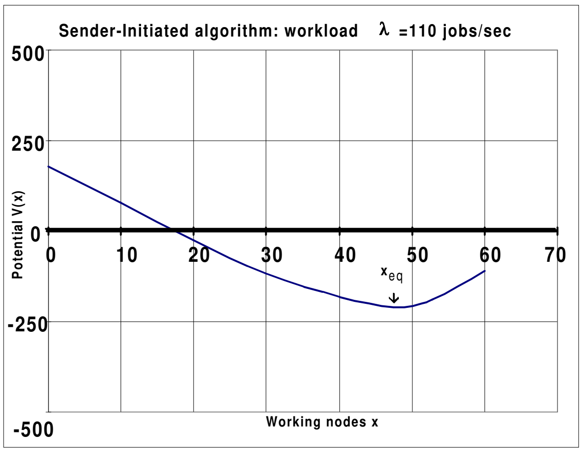


Figure 6.3

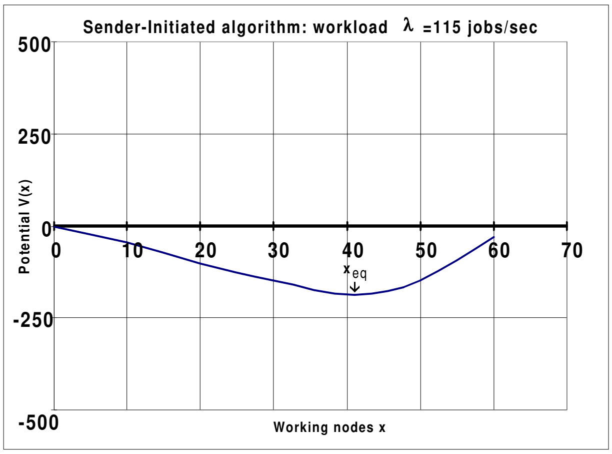


Figure 6.4

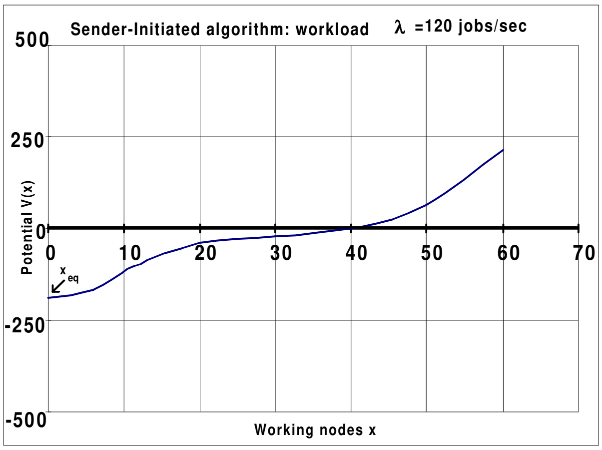


Figure 6.5

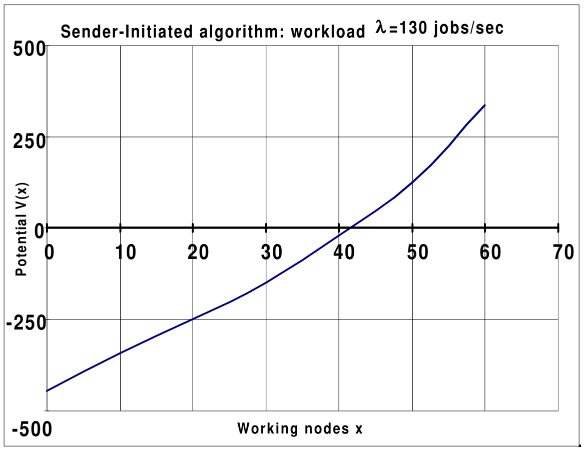


Figure 6.6

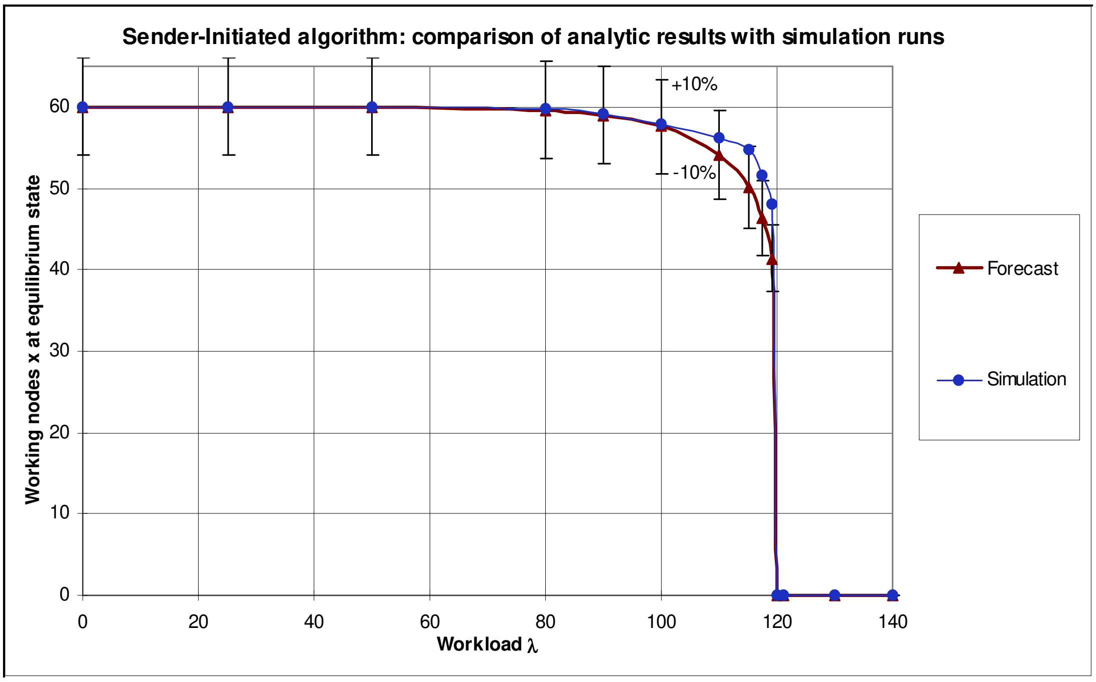


Figure 7

In order to evaluate the effect of the approximations made to obtain the analytical results for the "sender-initiated" algorithm, five simulations were run, using QNAP2. The following parameters have been used, where $D$ is the duration time of each simulation run:

$N = 60$ nodes, $\mu = 2$ job/sec, $\tilde{s} = 10$ jobs, $T = 20$ sec, $D = 2000$ sec

A comparison of analytical and simulation results is plotted in Fig. 7. In all the simulation runs, the discrepancy between analytical predictions and simulation results are localized for values of $\lambda$ in the proximity of the limit value $N{\cdot}\mu$. The observed error is always within 10 % and it is conservative for the analytical results. This discrepancy can be explained since, while the analytical model is based on the flow approximation, an M/M/1 queue model is used in the simulations.

## 6 Conclusions

In this work we analysed stability of different load balancing algorithms in a DDB with soft real-time transactions. The examples want to attest the utility of modelling computer systems using Catastrophe theory. Comparison with the simulation results shows that, for the model we studied, the error introduced using fluid approximation is conservative and quite small.

The Catastrophe theory can be useful to understand if an algorithm has a stable behaviour before it is implemented; in our case it was used to establish the maximum workload that a DDB can accept before transaction response time becomes unacceptable, or to dimension the processing nodes on the basis of the maximum transactions arrival rate foreseen to the DDB.

Moreover the method is simple to use: Potential function is the main notion to understand, and it is a well known concept of Physics.

Extensions of this work can lead in the direction of multidimensional problems, that is using more than one state variable, and this would allow one to analyze more complex and sophisticated models.

## 7 References:


[1] M. Baiguera, G. Bortolotto - "Valutazione della stabilità di algoritmi di riallocazione del carico in basi di dati distribuite, utilizzando la teoria delle catastrofi" - tesi di laurea (graduation thesis), Politecnico di Milano, (december 1995)

[2] G. C. Buttazzo, J. A. Stankovic - "Adding Robustness in Dynamic Preemptive Scheduling" - in "Responsive Computing Systems: Steps Toward Fault-Tolerant Real-Time Systems", D.S. Fussell , M. Malek (Eds.), Kluwer Academic Publ., Boston, (1995)

[3] E.L. Eager, E.D. Lazowska, J. Zahorian - "Adaptive Load Sharing in Homogeneous Distributed Systems" - IEEE-TSE, vol. SE-12, n. 5, pp.662-675, (1986)

[4] E.L. Eager, E.D. Lazowska, J. Zahorian - "A Comparison of Receiver-initiated and Sender-initiated Adaptive Loading" - Performance Evaluation, Vol. 6, (1986)

[5] D. Ferrari, S. Zhou - "An Empirical Investigation of Load Indices for Load Balancing Applications" - Proc. Performance'87, North Holland, pp. 515-528, (1988)

[6] A. Hac, X. Jin - "Dynamic Load Balancing in a Distributed System Using a Sender-Initiated Algorithm" - J. Systems Software, (1990)

[7] L. Kleinrock - "Queueing Systems Vol. 1" - John Wiley & Sons, (1976)

[8] R. Nelson - "The Stochastic Cusp, Swallowtail, and Hyperbolic Umbilic Catastrophes as Manifest in a Simple Communications Model" - Proc. Performance84, North-Holland, (1984)

[9] R. Nelson - "Stochastic Catastrophe Theory in Computer Performance Modelling" - Journal of the ACM, vol. 34, n. 3, pp. 661-685, (1987)

[10] T. Poston, I. Stewart - "Catastrophe Theory and Its Application" - Pitman, (1978)

[11] S. Rotenstreich - “Reliability of Distributed Programs under Overloads” - Journal of Systems and Software, vol. 9, pp. 29-40, (1989)

[12] F.A. Schreiber - “State Dependency Issues in Evaluating Distributed Database Availability” - Computer Networks, vol. 8, n. 3, pp. 187-197, (1984)

[13] F.A. Schreiber - “Notes on Real-Time Distributed Database System Stability” - Proc. JCIT-5, Jerusalem, IEEE Press, (1990)

[14] F.A. Schreiber, V. Caglioti - “Stabilization of Workload Distribution in Soft-Real Time Distributed Systems” - Tech. Report, Dipartimento di Elettronica e Informazione, Politecnico di Milano, (March 1994)

[15] B.A. Shirazi, A.R. Hurson, K.H. Kavi (Eds.) - “Scheduling and Load Balancing in Parallel and Distributed Systems” - IEEE- CS Press, (1995)

[16] J.A. Stankovic - “Stability and Distributed Scheduling Algorithms” - IEEE-TSE, vol. 11, n. 10, pp. 1141-1152, (1985)

[17] T. Sweeny, N. Sandler - “Bezeq Seeks Answers after Network Crash” - Communications Week International, n° 153, p. 24 ( Oct.1995)

[18] A. Thomasian - “Two Phase Locking and Its Thrashing Behavior” - ACM TODS, v. 18, n. 4, pp.579-625, (1993)

[19] S. Zhou - “An Experimental Assessment of Resource Queue Lengths as Load Indices” - Proc. Winter USENIX Conf., Washington, pp.73-82, (1987)