SymbioticSphere: A Biologically-Inspired Autonomic Architecture for Self-Adaptive and Self-Healing Server Farms

Paskorn Champrasert and Junichi Suzuki
Department of Computer Science
University of Massachusetts, Boston
{paskorn, jxs}@cs.umb.edu

Abstract

This paper describes a biologically-inspired architecture, called SymbioticSphere, which allows large-scale server farms to autonomously adapt to dynamic environmental changes and survive partial system failures. SymbioticSphere follows biological principles such as decentralization, autonomy, natural selection, emergence and symbiosis to design server farms (application services and middleware platforms). Each application service and middleware platform is designed as a biological entity, analogous to an individual bee in a bee colony. Simulation results show that, like in biological systems, SymbioticSphere exhibits emergence of desirable system characteristics such as adaptability and survivability.

1. Introduction

Server farms such as Internet data centers and grid clusters have become integral components to operate Internet services and scientific computation. Since they are rapidly increasing in complexity and scale, they face critical challenges, particularly *adaptability* and *survivability*. Server farms are expected to autonomously adapt to dynamic environmental changes such as demand surges and resource exhaustion [1]. They are also expected to autonomously survive partial system failures due to, for example, errors by administrators and physical damage to server farm fabric as a result of unexpected events (e.g., hurricanes and blackouts) [2].

Based on the observation that various biological systems have already developed necessary mechanisms to overcome the above challenges (i.e., adaptability and survivability), the proposed architecture, called SymbioticSphere, applies biological principles to design server farms (application services and middleware platforms). We believe if server farms adopt certain biological principles, they may be able to overcome these challenges.

In SymbioticSphere, each application service and middleware platform is modeled as a biological entity, analogous to an individual bee in a bee colony. Application services and middleware platforms are designed to follow several biological principles such as decentralization, autonomy, natural selection, emergence and symbiosis. An application service is designed as a software agent. Each agent implements a functional service and biological behaviors such as energy exchange, replication, death and migration. A middleware platform runs on a network host and operates agents. Each platform provides runtime services that agents use to perform their services and behaviors, and implements biological behaviors such as health monitoring, energy exchange, replication and death.

This paper describes and evaluates SymbioticSphere their impacts on adaptability and survivability of server farms. Simulation results demonstrate that agents and platforms autonomously adapt to dynamic environmental changes (e.g., demand surges) and survive host failures to retain their performance. Simulation results also show that agents and platforms spontaneously cooperate in certain circumstances to pursue their mutual benefits and improve their adaptability and survivability.

2. Design Principles in SymbioticSphere

SymbioticSphere consists of two components: agents and middleware platforms. Agents run on platforms, which in turn run on network hosts. Agents and platforms are designed based on the following principles.

(1) Decentralization: There are no central entities to control and coordinate agents/platforms (i.e., no directories and no resource managers). Decentralization allows agents/platforms to be scalable and survivable by avoiding a single point of performance bottlenecks and failures.

(2) Autonomy: Agents and platforms sense their local network environments, and based on the sensed environmental conditions, they autonomously behave, and interact with each other without any intervention from/to other agents, platforms and human users/administrators.

(3) Natural selection: Agents and platforms store and expend *energy* for living. Each agent gains energy in exchange for performing its service to other agents or human users, and expends energy to use network and computing resources. Each platform gains energy in exchange for providing resources to agents, and periodically evaporates energy. The abundance or scarcity of stored energy triggers natural selection of agents/platforms. For example, an abundance of stored energy indicates higher demand for an agent/platform; thus the agent/platform replicates itself to increase its availability. A scarcity of stored energy (an indication of lack of demand) causes death of the agent/platform. Like in biological natural selection where more favorable species in an environment becomes more abundant, the population of agents/platforms dynamically changes based on the demands for them.

(4) Emergence: Agents and platforms behave against dynamic environmental conditions (e.g. user demands and resource availability). For example, an agent may invoke migration behavior to move towards a platform that forwards a large number of request messages for its services. Also, a platform may replicate itself on a neighboring host whose resource availability is high. Through collective behaviors and interactions of individual agents and platforms, desirable system characteristics such as adaptability and survivability emerge in a swarm of agents and platforms. Please note that the desirable characteristics are not present in any single agent/platform.

(5) Symbiosis: Agents and platforms are modeled as different species. In certain circumstances, agents and platforms spontaneously cooperate to pursue their mutual benefits and improve their adaptability and survivability, although each of them is not explicitly designed to do so.

3. SymbioticSphere

This section describes the design of SymbioticSphere.

3.1 The Architecture of SymbioticSphere

Fig. 1 shows the architecture of SymbioticSphere. Agents and platforms are modeled as different biological species. As a living entity, the ultimate goal of each species is to survive for a long time by balancing its energy gain and population. SymbioticSphere follows ecological principles to design energy exchange among agents, platforms and environment. It models a user as the Sun, agents as producers, and platforms as consumers[1]. Similar to the Sun, users have unlimited amount of energy. Each agent gains energy from users[2] and transfers 10% of its energy level to an underlying platform for consuming resources provided by the platform. Each platform gains energy from agents and evaporates 10 % of its energy level to the environment. This energy exchange rule follows an ecological fact that about 10% of the energy maintained by producer species goes to consumer species [3]. Due to space limitation, see [4] for more details on energy exchange in SymbioticSphere.

3.2 Agents

Each agent consists of three parts: *attributes*, *body* and *behaviors*. *Attributes* carry descriptive information regarding the agent, such as agent ID, energy level and description of a service it provides. *Body* implements a service that the agent provides. For example, an agent may implement a web service, while another agent may implement a physical model for scientific simulations. *Behaviors* implement actions that are inherent to all agents. Although SymbioticSphere defines nine standard agent behaviors [5], this paper focuses on three of them.

[1] In the ecological system, producers (e.g., shrubs) convert the Sun light energy to chemical energy. The chemical energy is transferred to consumers (e.g., hares) as consumers consume producers.

[2] Each agent specifies the price (in energy units) of its service.

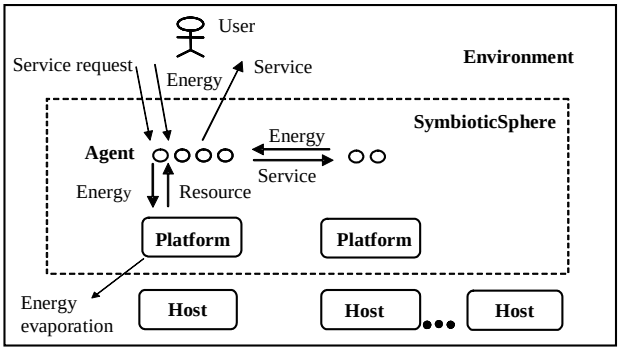

Fig. 1 Energy Exchange in SymbioticSphere

- *Replication*: Agents may make a copy of themselves as a result of abundance of energy. A replicated (child) agent is placed on the platform that its parent agent resides on, and it receives the half amount of the parent's energy level.
- *Death*: Agents die due to energy starvation. When an agent dies, an underlying platform removes the agent and releases all resources allocated to the agent.
- *Migration:* Agents may move from one platform to another.

3.3 Platforms

Each platform runs on a network host and operates agents. It consists of *attributes*, *behaviors* and *runtime services*. *Attributes* carry descriptive information regarding the platform, such as platform ID, energy level and health level. Health level indicates how healthy an underlying host is. It is defined as a function of three properties: resource availability on, age of and freshness of a host. Resource availability indicates how much resources are available for agents and platforms on a host. Age indicates how long a host has been alive (i.e., how much stable a host is). Freshness indicates how recently a host joined the network. After a new host joined the network, its freshness gradually decreases from the maximum value. When an unstable host resumes from a failure, its freshness starts with the value that the host has when it went down. Using age and freshness, unstable hosts and new hosts can be distinguished (Table 1). Health level affects behaviors of each platform and agent. For example, higher health level indicates higher stability of and/or higher resource availability on a host that a platform resides on. Thus, the platform may replicate itself on a healthier neighboring host.

Table 1.Freshness and Age in Different Types of Hosts

Host Type	Freshness	Age
Unstable host	Lower	Lower
New host	Higher	Lower
Stable host	Lower	Higher

Behaviors are the actions inherent to all platforms.

- *Replication.* Platforms may make a copy of themselves as a result of abundance of energy (i.e., higher demand

for resources available on the platforms). The child platform inherits the half of the parent's energy level.

- *Death.* Platforms die due to lack of energy. A dying platform uninstalls itself and releases all resources the platform uses. Despite the death of a platform, an underlying host remains active so that another platform can run on it in the future.

Runtime services are middleware services that agents and platforms use to perform their behaviors.

3.4 Behavior Policies of Agents and Platforms

Each agent/platform has policies for its behaviors. A behavior policy defines when to and how to invoke a particular behavior. Each behavior policy consists of *factors* (F_i), which evaluate environment conditions (e.g. network traffic) or agent/platform/host status (e.g. energy level and health level). Each factor is given a *weight* (W_i) relative to its importance. Behaviors are invoked if the weighted sum of factor values ($\Sigma F_i * W_i$) exceeds a threshold.

The factors in agent migration behavior policy include:

- *Energy Level:* Agent energy level, which encourages agents to move in response to higher energy level.
- *Health Level Ratio*: The ratio of health level on a remote host to the local host, which encourages agents to move to platforms running on healthier hosts. This ratio is calculated with three health level properties (i.e., resource availability, freshness or age) as follows:

$$Health\ Level\ Ratio = \sum_{i}^{3}\left(\frac{HealthLevel\,\mathrm{Pr}operty_i on\ remote\ host - HealthLevel\,\mathrm{Pr}operty_i on\ local\ host}{HealthLevel\,\mathrm{Pr}operty_i on\ local\ host}\right) \quad \text{... (1)}$$

- *Service Request Ratio*: The ratio of # of incoming service requests on a remote platform to the local platform, which encourages agents to move towards users.
- *Migration Interval*: Time interval to perform migration, which discourages agents to migrate too often.

If there are multiple neighboring platforms that an agent can migrate to, the agent calculates a weighted sum of the above factors for each of the platforms, and moves to a platform that generates the highest weighted sum.

The factors in agent replication behavior policy include:

- *Energy Level:* Agent energy level, which encourages agents to replicate themselves in response to higher energy level.
- *Request Queue Length:* The length of service request queue, which the local platform maintains to queue incoming service requests. This factor encourages agents to replicate themselves in response to higher demands.

The factors in agent death behavior policy include:

- *Energy Level:* Agent energy level. Agents die when they run out of their energy.
- *Energy Loss Rate*: The rate of energy loss in between the current and previous simulation cycles. This factor is calculated with the following equation, where E_t and E_{t-1} are the energy levels in the current and previous simulation cycles. Agents have higher risk to die in response to sharp drop in demands for their services.

$$Energy\ \ Loss\ \ Rate = \frac{E_{t-1} - E_t}{E_{t-1}} \quad \text{..... (2)}$$

The factors in platform replication behavior include:

- *Energy Level:* Platform energy level, which encourages platforms to replicate themselves in response to higher energy level.
- *Health Level Ratio*: The ratio of health level on a remote host to the local host, which encourages platforms to replicate themselves on healthier neighboring hosts. This ratio is calculated with Equation (1).
- *The Number of Agents:* The number of agents working on each platform. This factor encourages platforms to replicate themselves in response to higher agent population on them.

If there are multiple neighboring hosts that a platform can replicate itself on, the platform places a child platform on a host whose health ratio is highest among others.

The factors in platform death behavior include:

- *The Number of Agents*: The number of agents running on each platform. This factor discourages platforms to die when agents run on them.
- *Energy Loss Rate:* The rate of energy loss in platforms. This factor is calculated with Equation 2. Platforms have higher risk to die in response to sharp drop in demands for their resources.

Each agent/platform expends energy to invoke behaviors (i.e., behavior cost) except death behavior. When the energy level of an agent/platform goes over the cost of a behavior, the agent/platform decides whether it performs the behavior by calculating a weighted sum of factors.

4. Simulation Results

This section shows simulation results to evaluate how biologically-inspired mechanisms in SymbioticSphere impact adaptability and survivability of server farms[3]. Fig. 2 shows a pseudo code of each simulation cycle.

```
While (not the last simulation cycle)
    For each user do
        send service requests to each of available agents according to
        a configured service request rate.
    End For
    If (simulation cycle mod an interval = 0) do
        For each platform do
            make a decision on performing a behavior.
            update healthy level.
            expend (evaporate) energy.
        End For
    End If
    For each agent do
        While (not exceeds the max # of msgs an agent can process) do
            If (a service request(s) arrived) do
                process the request(s) and gain energy.
            End If
        End While
        If (simulation cycle mod an interval = 0) do
```

[3] Simulations were carried out with the SymbioticSphere simulator, which contains 14,600 lines of Java code.

```
            make a decision on performing a behavior.
            transfer energy to the local platform.
        End If
    End For
End While
```

Fig. 2 Pseudo Code of Simulation Cycle

Fig. 3 shows a simulated network. A server farm consists of hosts connected in a *N* x *N* grid topology, and users send service requests to agents via user access point. This paper assumes that a single (virtual) user runs on the access point and it emulates multiple users to send service requests. Each host has 320 MB or 256 MB memory space[4]. Out of the space, an operating system and Java VM consume 128 and 64 MB, respectively. The remaining space is available for a platform and agents on each host. Each agent and platform consumes 5 and 20 MB, respectively. This assumption is obtained from a prior empirical experiment [5].

Each simulation runs for 24 hours in simulation time.

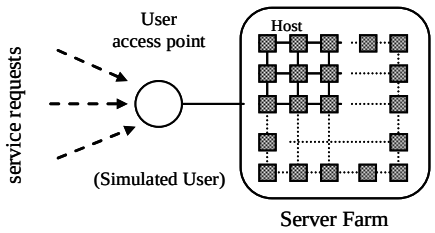

Fig 3 Simulated Network

4.1. Adaptability of SymbioticSphere

The first simulation study is carried out to evaluate the adaptability of SymbioticSphere. Adaptability is evaluated as *service adaptation* and *resource adaptation*. *Service adaptation* is the activities to adaptively improve the quality and availability of services provided by agents. Quality of service is measured as response time of agents to respond service requests from users. Service availability is measured as the number of available agents. *Resource adaptation* is the activities to adaptively improve resource availability and resource efficiency. Resource availability is measured as the number of platforms that make resources available for agents. Resource efficiency indicates how many service requests are processed per resource utilization.

Service request rate starts with 3,000 requests/min, spikes to 210,000 requests/min at 8:00, and drops to 3,000 requests/min at 16:30 (Fig. 5). The peak demand and spike ratio (1:70) are taken from a workload trace of the 1998 World Cup web site [6]. A simulated server farm is 7x7 (49 hosts) from 0:00 to 12:00 and 15x15 (225 hosts) from 12:00 to 24:00. 30% of the hosts (resource rich hosts) have 320MB memory space. The other 70% are resource poor hosts, which have 128MB. Both types of hosts are placed randomly. At the beginning of a simulation, a single agent and platform is deployed on each host.

[4] Currently, memory availability represents resource availability.

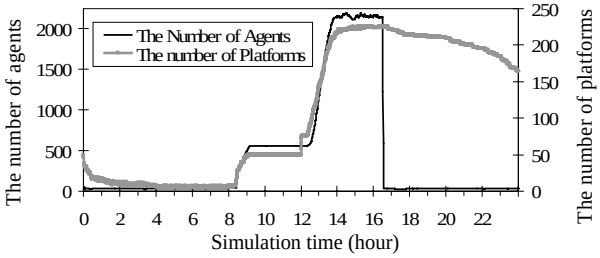

Fig 4 The Number of Agents and Platforms

Fig. 4 shows how service availability (i.e., the number of agents) and resource availability (i.e., the number of platforms) change dynamically. Starting with 49 agents and 49 platforms at 0:00, they autonomously adapt their populations to demand changes. When service request rate spikes at 8:00, agents gain more energy form users and replicate themselves more often. In response to higher energy intake, they also transfer more energy to platforms. As a result, platforms also increase their population through replications. From 10:00 to 12:00, the populations of agents and platforms do not grow due to physical limitation of available hosts. When the size of a server farm expands at 12:00, agents and platforms recognize the environmental change and rapidly increase their populations. When service request rate drops at 16:30, most agents die because they cannot balance their energy gain and expenditure. Also, the population of platforms gradually decreases due to less energy transfer from agents. Fig. 4 shows that biological mechanisms contribute for agents and platforms to adaptively adjust their availability to dynamic demand changes.

Fig. 5 shows the rate of service requests from users and the throughput achieved by agents. It depicts that agents and platforms collectively adapt throughput performance to dynamic changes in demand (8:00 and 16:30) and server farm size (12:00). Fig. 5 also shows that the biological mechanisms in SymbioticSphere scales well to demand volume, spike ratio and server farm size.

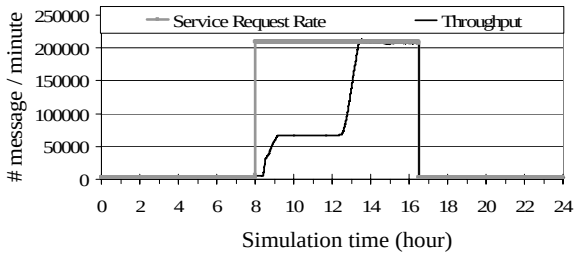

Fig. 5 Service Request Rate and Throughput

Fig. 6 shows the quality of service (i.e., the average response time for agents to respond users). This includes the message transmission latency between a user and agent and the processing overhead for an agent to process a service request. From 0:00 to 8:00, response time decrease because agents migrate towards users. At 8:00, response time spikes because service request rate spikes. Response time starts decreasing at 12:00 when agents/platforms

replicate on the hosts that newly joins the server farm. At 18:00, response time spikes because some agents die near from users due to energy starvation caused by a drop in service request rate. Fig. 6 shows the biological mechanisms in SymbioticSphere contribute for agents and platforms to keep response time low despite demand surges.

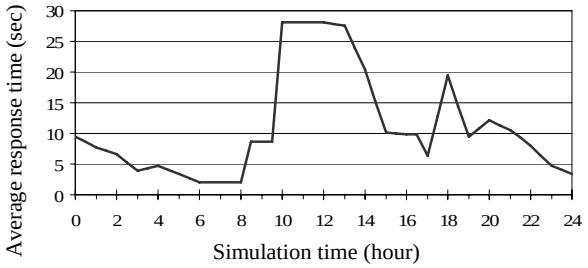

Fig. 6 Average Response Time

Fig. 7 shows resource utilization balancing index

$$Resource\ Utilization\ Balancing\ Index = \sqrt{\frac{\sum_{i}^{N}(R_i - \mu_R)^2}{N}} \quad \text{..... (3)}$$

(RUBI). RUBI is measured with Equation 3.

R_i represents resource utilization rate on the host *i*: (the amount of resources a platform and agents utilize on the host *i*) / (the total amount of resources the host *i* has). *N* is the number of hosts that platforms reside on. μ_R represents the expected average of resource utilization rate on *N* hosts. The lower RUBI is, the more resource utilization is distributed over hosts in proportion to the amount of resources the hosts have. This means that more agents and platforms run on resource righ hosts, and less agents run on resource poor hosts. From 0:00 to 8:00, RUBI gradually decrease. This means that agents move to and platforms replicate themselves on resource rich (i.e., healthier) hosts. At 8:00, RUBI increases because more agents and platforms start running on both resource rich and poor hosts when service request rate spikes. After that, agents and platforms seek resource rich hosts through migration and replication, resulting in lower RUBI. At 12:00, RUBI spikes because agents and platforms rapidly spread over both resource rich and poor hosts to handle high service request rate when new hosts join the server farm. However, agents and platforms decrease RUBI again by preferentially residing on resource rich hosts. Fig. 7 shows that the biological mechanisms in SymbioticSphere contribute for agents and platforms to adaptively balance resource utilization over heterogeneous hosts.

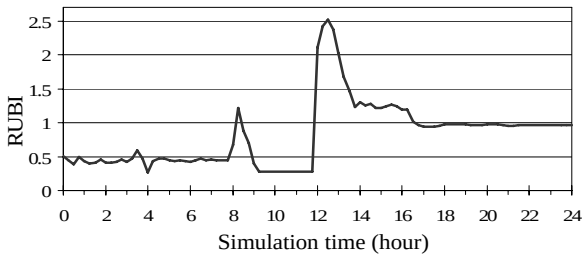

Fig. 7 Resource Utilization Balancing Index (RUBI)

Fig. 7 also shows an example of symbiotic emergence between agents and platforms. Agent migration behavior policy encourages agents to move towards platforms on healthier hosts. Platform replication behavior policy encourages platforms to replicate themselves on healthier hosts. Therefore, when new hosts join a server farm at 12:00, platforms perform replications on the new hosts and agents migrate to the replicated platforms. As a result, service requests are processed by agents that are spread over the platforms running on healthy hosts. This contributes to balance workload and resource utilization on each platform, although agent migration policy and platform replication policy do not consider agent population, platform population, load balancing, and resource utilization balancing. This results in a mutual benefit for both agents and platforms. Platforms help agents decrease response time by making more resources available for them. Agents help platforms to keep their stability by avoiding excessive resource utilization on them.

Fig. 8 shows resource efficiency, which indicates how many service requests are processed per resource utilization. It is measured as (the total # of service requests processed by agents) / (the total amount of resources consumed by all agents and platforms). Until 8:00, agents and platforms keep increasing resource efficiency by adjusting their populations to service demand and decreasing response time for users (see Figs. 4 and 6). Fig. 8 shows biological mechanisms in SymbioticSphere contribute for agents and platforms to adaptively improve resource efficiency by using available resources effectively.

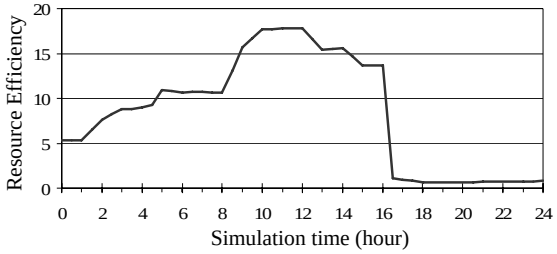

Fig. 8 Resource Efficiency

4.2. Survivability of SymbioticSphere

The second set of simulations is carried out to evaluate the survivability of SymbioticSphere. Service request rate is constantly 7,200 requests/min, which is taken from a workload trace of the IBM web site in 2001 [7]. The server farm size is 7x7 (49 hosts). Each host has 256MB memory space. In each simulation, randomly chosen 30% or 60% of hosts go down at 9:00 for 90 minutes.

Figs. 9 and 10 show how resource availability (i.e., the number of platforms) and service availability (i.e., the number of agents) dynamically change, respectively. When hosts go down at 9:00, agents and platforms die on the failed hosts, resulting in a drop in agent/platform population. However, after the host failure, agents and platforms replicate themselves on remaining hosts in or-

der to keep (or increase) their populations. Therefore, agents improve their throughput during the period of host failure (Fig. 11). Figs. 9, 10 and 11 show that biological mechanisms in SymbioticSphere contribute for agents and platforms to increase their populations and increase agent throughput despite significant host failures.

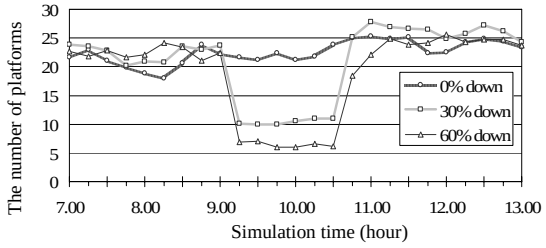

Fig. 9 The Number of Platforms

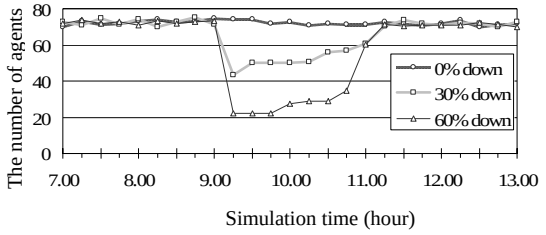

Fig. 10 The Number of Agents

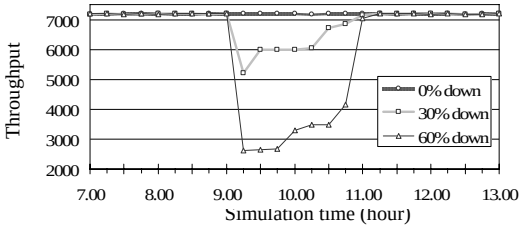

Fig. 11 Throughput

5. Related Work

The Bio-Networking Architecture [5] proposes biologically-inspired agents to achieve service adaptation in a decentralized and collective manner. However, platforms do not achieve resource adaptation because they are static and non-biological entities. In SymbioticSphere, both agents and platforms are biological entities, and they achieve service adaptation and resource adaptation simultaneously. SymbioticSphere also exhibits a new from of adaptation, symbiotic emergence, which does not appear in the Bio-Networking Architecture.

Resource Broker [8] and Muse [7] achieve resource adaptation for server clusters via centralized system monitor. Resource Broker inspects the stability and resource availability of each host, and adjusts resource allocation for applications. Muse inspects electric power consumption of a server cluster and adjusts resource allocation for applications. Rather than following centralized architectures, SymbioticSphere achieves both service adaptation and resource adaptation with decentralized agents and platforms. Also, Muse does not consider survivability against host failures.

Rainbow achieves both service adaptation and resource adaptation in server clusters [9]. A centralized system monitor inspects the current environment conditions, and performs an adaptation strategy (e.g. service migration and platform replication/death). SymbioticSphere implements more adaptation strategies such as agent replication and agent death. It also addresses survivability as well as adaptability with the same set of agent/platform behaviors. Rainbow does not consider survivability from failures.

[10] proposes a decentralized design for server clusters to guarantee response time. SymbioticSphere does not guarantee any system measures including response time because the dynamic improvement of those measures is an emergent result from collective behaviors and interactions of agents and platforms. As a result, agents and platforms can adapt to unexpected environmental changes (e.g., system failures) and survive them without changing any behaviors and their policies. [10] does not consider survivability from system failures.

6. Conclusion

This paper overviews SymbioticSphere, and presents how it implements biological mechanisms to improve the adaptability and survivability of server farms. Simulation results show SymbioticSphere allows server farms to autonomously adapt to dynamic environmental changes and survive partial system failures.

Reference

[1] J. Rolia and S. Singhal and R. Friedrich, "Adaptive Internet Data Centers," *Proc. SSGRR'00*, July 2000.

[2] A. Nguyen-Tuong, A. S. Grimshaw, G. Wasson, M. Humphrey, J.C. Knight, "Towards Dependable Grids," University of Virginia, TR-CS-2004-11, 2004.

[3] R. M. Alexander, "Energy for Animal Life," Oxford University Press, May 1999.

[4] P. Chaprasert and J. Suzuki, "SymbioticSphere: A Biologically-inspired Network Architecture for Autonomic Grid Systems," *Proc. of IASTED CIIT*, October 2005.

[5] J. Suzuki and T. Suda, "A Middleware Platform for a Biologically-inspired Network Architecture Supporting Autonomous and Adaptive Applications" *IEEE J. on Selected Areas in Comm.* Feb. 2005.

[6] M.F. Arlitt and T. Jin, "A Workload Characterization Study of the 1998 World Cup Web Site," *IEEE Network*, May/June 2000

[7] J. Chase, D. Anderson, P. Thakar, and A. Vahdat, "Managing Energy and Server Resources in Hosting Centers," *Proc. of ACM SOSP*, October 2001

[8] A. Othman, P. Dew, K. Djemame, I, Gourlay, "Adaptive Grid Resource Brokering," *Proc. of IEEE Cluster*, Dec. 2003.

[9] S. Cheng, D. Garlan, B. Schmerl, P. Steenkiste, and N. Hu, "Software Architecture-based Adaptation for Grid Computing," *Proc. of IEEE HPDC*, July 2002.

[10] C. Adam, R. Stadler, "Adaptable Server Clusters with QoS Objectives," *Proc.* of IFIP/IEEE IM, May, 2005.