Towards Green Grids: A Biologically-Inspired Adaptive Architecture for Power Efficient 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 network architecture, called SymbioticSphere, which allows server farms to autonomously adapt to dynamic environmental changes and achieve power efficiency. SymbioticSphere follows certain biological principles such as decentralization, natural selection, emergence and symbiosis to design server farms (application services and middleware platforms). Each application service and middleware platform is modeled as a biological entity, analogous to an individual bee in a bee colony. Simulation results show that, like in biological systems, desirable system properties (e.g., adaptability and power efficiency) emerge from collective actions and interactions of application services and platforms.

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 have been increasing in scale and complexity, server farm electricity costs are already in the range of $3.3 billion annually (in 2005), and the number of servers in the United States is expected to increase more than 50% over the next four years [1]. [2] reports that 2% of electricity consumption in the US (74TWh/year; $6 billion/year) was used to operate the Internet in 1999. According to [3], the US Secretary of Energy reported that the demands of the Internet already consume 8% to 13% of electricity in 2001. As such, one of the current key issues for server farm administrators is to balance electricity costs and computation power.

Regarding the workload placed on server farms, the recent research efforts have revealed large daily and seasonal fluctuations in workload [4, 5, 6]. Therefore, this paper studies an autonomous and adaptive strategy in which server farms autonomously achieve power efficiency through adapting their operations to dynamic environmental changes in the network (e.g., changes in workload and resource availability).

In order to make this power efficiency strategy a reality, this paper proposes a network architecture, called SymbioticSphere, which applies biological principles to design autonomous and adaptive server farms with power efficiency in mind. SymbioticSphere is motivated by the observation that various biological systems have developed the mechanisms necessary to achieve autonomy and adaptability. For example, bees in a bee colony act autonomously, influenced by local conditions and local interactions with other bees. A bee colony adapts to dynamic environmental conditions. When the amount of honey in a hive is low, many bees leave the hive to gather nectar from flowers. When the hive is full of honey, most of bees rest in the hive. Bees also adjust energy consumption in different seasons by changing reproduction rate [7]. We believe that if server farms adopt certain biological principles, they may be able to attain autonomy and adaptability for power efficiency.

SymbioticSphere consists of application services and middleware platforms. Each of them is modeled as a biological entity, analogous to an individual bee in a bee colony. Both application services and middleware platforms follow several biological principles such as decentralization, natural selection, emergence and symbiosis. An application service is implemented as an autonomous software agent. Each agent implements a functional service and follows simple biological behaviors such as replication, death, migration and energy exchange. A middleware platform runs on a network host and operates agents. Each platform provides a set of runtime services that agents use to perform their services and behaviors, and implements simple biological behaviors such as replication, death and energy exchange.

This paper describes the biologically-inspired mechanisms in SymbioticSphere and evaluates how they contribute for server farms (i.e., agents and platforms) to autonomously improve their power efficiency through adapting to dynamic environmental changes. Simulation results show server farms autonomously adapt to dynamic network environment and save over 50% of power consumption without sacrificing their performance.

This paper is organized as follows. Section 2 summarizes the key design principles in SymbioticSphere. Section 3 describes the design of agents and platforms. Section 4 evaluates simulation results. Sections 5 and 6 conclude with discussion on related work.

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 simple by avoiding a single point of performance bottlenecks [8] and avoiding any central coordination to deploy agents/platforms [9].

(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.

(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. 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. Note that these desirable characteristics are not present in any single agent/platform.

(5) Symbiosis: Agents and platforms are modeled as different biological species. The two types of biological entities are designed to complement with each other (or in a symbiotic manner). Agents cannot survive without platforms, and platforms cannot survive without agents. This symbiotic relationship between agents and platforms can improve power efficiency by turning off the network hosts that host no agents.

3. SymbioticSphere

This section presents the design of SymbioticSphere.

3.1 The Architecture of SymbioticSphere

SymbioticSphere models agents and platforms as different species, and follows ecological principles to design energy exchange among agents, platforms and environment. Fig. 1 shows a simplified energy flow in the ecological system. The Sun gives light energy, and producers (e.g., plants and microorganisms) convert it to chemical energy. The chemical energy flows through multiple species, called consumers. It will be eventually transferred to decomposers (e.g., bacteria and fungi). For example, shrubs (producers) convert the Sun light energy to chemical energy, hares (primary consumers) consume shrubs, and foxes (secondary consumers) consume hares.

In energy exchange between different species, it is known that about 10% of the energy maintained by one species is transferred to another species [10]. The remaining 90% of the energy is used for metabolism, growth and actions/behaviors (e.g., reproduction).

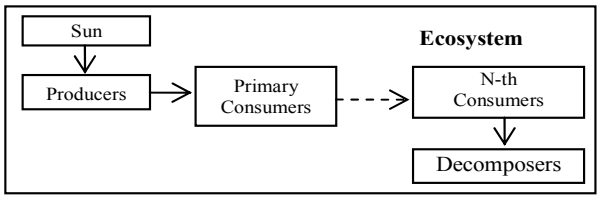

Fig. 1 Energy Flow in Ecosystem

Fig. 2 shows the energy exchange in SymbioticSphere. SymbioticSphere models each user as the Sun, agents as producers and platforms as (primary) consumers. Similar to the Sun, users have unlimited amount of energy. Agents gain energy from users[1], and expend energy to consume resources provided by platforms (e.g., memory space). They periodically transfer 10% of the current energy level to platforms on which they operate. Platforms periodically gain energy from agents, and evaporate 10 % of the current energy level to the environment.

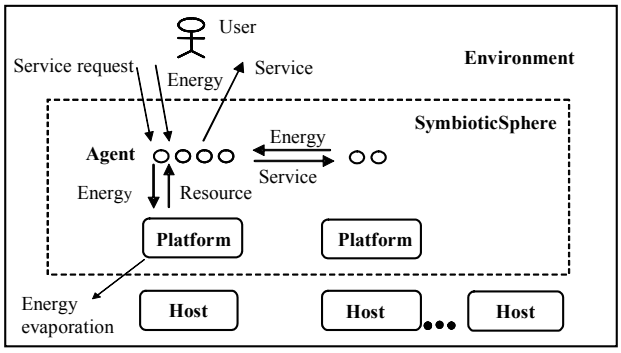

Fig. 2 Energy Exchange in SymbioticSphere

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

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

may implement a web service and contains web pages in its body 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 [11], this paper focuses on three of them.

- *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* is defined as a function of the age of and resource availability on a network host that the platform runs on. The age indicates how long a network host remains alive (i.e., how much stable a network host is). Resource availability indicates how much resources (e.g., memory space) are available for agents and platforms on a network host. Health level affects behaviors of a platform and agent. For example, higher health level indicates higher stability of and/or higher resource availability on a network host that the platform resides on. Thus, the platform may replicate itself on a healthier neighboring host than the current local host. This results in the adaptation of platform locations. Platforms strive to run on more stable and resource rich network hosts.

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 receives the half amount of the parent's energy level.
- *Death.* Platforms die due to the lack of energy. A dying platform uninstalls itself and releases all resources the platform uses.

Runtime services are middleware services that agents and platforms use to perform their behaviors. In order to maximize decentralization and autonomy of agents/platforms, they only use their local runtime services. They are not allowed to invoke any runtime services running on a remote platform.

3.4 Behavior Policies of Agents and Platforms

Each agent and platform has policies for its behaviors. A behavior policy defines when to and how to invoke a particular behavior. Each behavior policy consists of one or more *factors* (F_i), which evaluate environment conditions (e.g., network traffic and resource availability) or the status of agent/platform/host (e.g., energy level and health level)[2]. Each factor is given a *weight* (W_i) relative to its importance. Behaviors are invoked if the weighted sum of factor values (ΣF_i*W_i) exceeds a threshold.

The factors in agent migration behavior include:

- *Health level Ratio*: The ratio of health levels in a remote and the local hosts. This factor encourages agents to move to platforms on healthier hosts.
- *Service Request Ratio*: the ratio of the number of service requests on a remote platform by the number of service requests on a local platform, which encourages agents to move towards users.
- *Migration interval*: interval from the time of a previous 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 platform, and move to a platform that generates the highest weighted sum.

Agent replication and death behaviors have a factor that evaluates the current energy level of agent.

Platform replication behavior has a factor of health level ratio, which encourages platforms to replicate themselves on a healthier neighboring host. A replicated (child) platform is placed on a host whose health level is highest among neighboring hosts.

Platform death behavior has a factor that evaluates the current energy level of platform. Platforms never die while an agent(s) runs on the platform.

Each agent/platform incurs energy loss to invoke behaviors (i.e., behavior cost) except death behavior. When the energy level of an agent/platform exceeds the cost of a behavior, the agent/platform decides whether it performs the behavior by calculating a weighted sum of factor values.

4. Simulation Results

This section shows simulation results to evaluate how the biologically-inspired mechanisms in SymbioticSphere impact the adaptability and power efficiency of server farms[3]. In this paper, adaptability is an ability of server farms (i.e., agents and platforms) to adjust the availability (service availability and resource availability) and performance (response time and throughput) according to dynamic environmental conditions. Service availability is measured as the number of available

[2] Each agent and platform can sense its surrounding environment conditions. An agent can sense agent population, network traffic and resource availability on the local and neighboring platforms. A platform can sense agent population on itself, and health level of the local and neighboring hosts.

[3] Simulations were carried out with the SymbioticSphere simulator, which contains 14,200 lines of Java code. It is available for researchers who investigate autonomic network systems (dssg.cs.umb.edu).

agents. Resource availability is measured as the number of platforms that make resources available for agents. Power efficiency is measured with the power consumption by network hosts.

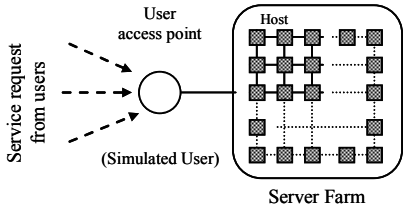

Fig. 3 Simulated Network

Fig. 3 shows a simulated network. A server farm consists of hosts connected in an N x N grid topology, and service requests travel from users to agents via user access point. This simulation study assumes that a single (emulated) user runs on the access point and sends service requests to agents. Each host has an Intel Pentium 4 CPU, 100Mbps Ethernet interface card and 256MB memory[4]. Out of the memory space, an operating system consumes 128 MB, and Java virtual machine consumes 64MB. Thus, 64MB is available for a platform and agents on each host. Each agent and platform consumes 5 MB and 20 MB, respectively. This assumption is obtained from a prior empirical experiment [11].

Each host operates in active or inactive state. When a platform works on a host, the host is in active state and consumes 60W power. The host goes to inactive state when a platform dies on it. An inactive host consumes 5W power. This assumption on power consumption is obtained from [12]. A host can become active from inactive state using the Wake On LAN (WOL) technology [13]. When a platform replicates itself on an inactive host, the platform sends a WOL packet to the host to wake it up.

Fig. 3 shows a pseudo code to run users, agents and platforms in each simulation cycle.

```
While (not the last cycle in a simulation)
    For each user Do
        Send service requests to agents according to a configured
        service request rate.
    End For
    For each platform Do
        Make a decision on replication and death behaviors.
        Update health level.
        Expend (evaporate) energy.
    End For
    For each agent Do
        If (a service request(s) arrived)
            Process the request(s) and gain energy.
        End If
        Make a decision on replication, migration and death behaviors.
        Expend energy to the local platform.
    End For
End While
```

Fig. 3 Pseudo Code of Each Simulation Cycle

When a user issues a service request, the service request is passed to the local platform where the user resides on, and the platform performs a discovery process to search a target agent that can process the issued service request. The platform (discovery originator) forwards a discovery message to its neighboring platforms, asking whether they host a target agent. If a neighboring platform hosts a target agent, it returns a discovery response to the discovery originator. Otherwise, it forwards the discovery message again to its neighboring platforms. Fig. 4 shows this peer-to-peer agent discovery process through platform connectivity[5].

```
While (not simulation last cycle)
  If ( Discovery messages arrived)
    For each of discovery messages (under the max # of messages to be
        processed in each simulation cycle) Do
        If ( Discovery message matches one of the local agents)
            Return a discovery response to discovery originator.
        Else
            Forward the discovery message to neighboring platforms.
        End If
    End For
  End If
End While
```

Fig. 4 Pseudo Code for Agent Discovery Process in Each Simulation Cycle

Through the above discovery process, a user finds a set of platforms hosting the target agents that can process his/her service request. The user chooses the platform closest from him/her and forwards his/her service request to the platform. When the service request arrives the platform, it inserts the request in its request queue. Each platform inspects the length of its request queue and the number of local agents running on it in each simulation cycle. If the number of queued service requests exceeds the number of service requests that local agents can process in a simulation cycle, the platform propagates the queued requests to other platforms hosting the agents that can process the requests. This propagation continues until the number of queued requests becomes is smaller than the number of service requests that local agents can process in a simulation cycle. Fig. 5 shows this request propagation process.

```
While (not the last cycle in a simulation)
  If ( # of queued service requests > # of service requests that local agents
      can process in a simulation cycle )
      If (there is available platform)
          transfer service requests to available hosts in round robin
      End If
  Else If
      Process the service request
  End If
End While
```

Fig. 5 Pseudo Code to Process and Propagate Service Requests in Each Simulation Cycle

Each simulation runs for 24 hours in simulation time. Fig. 6 shows how service request rate changes through a

[4] Currently, memory availability represents resource availability on each platform/host.

[5] Note that there is no centralized directory to keep track of agents.

simulation. This is taken from a workload trace of the 1998 Olympic official website [14]. (The peak is 9,600 request/min.) A simulated server farm is 7x7 (49 hosts). At the beginning of a simulation, one agent and one platform are deployed on a host.

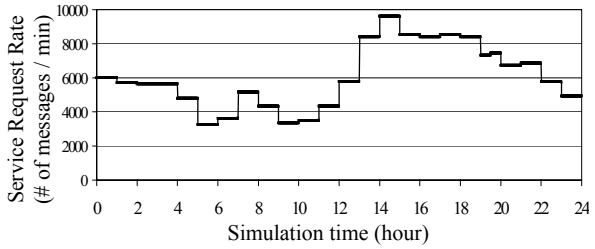

Fig. 6 Service Request

Fig. 7 shows how service availability (i.e., the number of agents) and resource availability (i.e., the number of platforms) change dynamically. Starting with an agent and a platform at 0:00, they change their populations through replication in order to handle the demand placed on them (6,000 requests/min). When service request rate increases from 12:00 to 2:00, 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. When service request rate decreases from 14:00, some of agents and platforms die because they cannot balance energy gain and expenditure due to less energy transfer from users. Fig. 7 shows that biological mechanisms in SymbioticSphere contribute for agents and platforms to autonomously adapt their availability to dynamic demand changes.

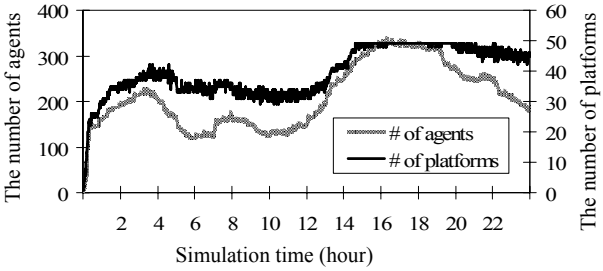

Fig. 7 The Number of Agents and Platforms

Fig. 8 shows the average response time and the throughput achieved by agents. In the first hour, response time is high (25 sec) because there is only one agent and one platform needs to process 6,000 requests a minute at the beginning of a simulation. As a result, throughput does not reach 100%. However, as agents and platforms replicate themselves and agents migrate towards users, the response time drops to 1 second at 2:00. (throughput reaches 100%.) After 2:00, the response time is constantly 1 second and the throughput is constantly 100%, although service request rate increases from 12:00 to 2:00. This means agents and platforms responsively change their populations and locations against demand changes. Fig. 8 shows that the biological mechanisms in SymbioticSphere contribute for agents and platforms to collectively retain response time and throughput performance by adjusting their populations and locations.

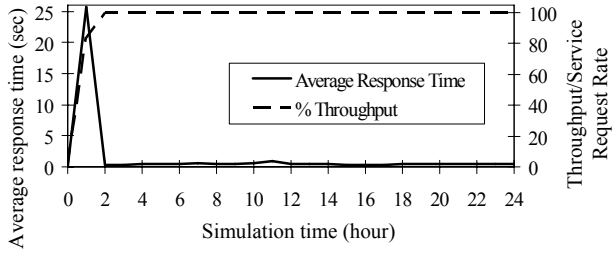

Fig. 8 Response Time and Throughput

Fig. 9 shows resource efficiency, which indicates how many service requests can be processed per resource unit. It is measured as (the total number of user requests processed by agents) / (the total amount resources consumed by agents and platforms). The resource efficiency of SymbioticSphere is compared with a scenario where platforms do not have biological behaviors (replication and death). This scenario simulates the always-on operation in traditional grids. (49 platforms always run on hosts.) Since platforms of SymbioticSphere dynamically adjust resource availability according to demand changes (Fig. 7), SymbioticSphere outperforms traditional grids in resource efficiency. Figs. 9 and 8 show that the biological mechanisms in SymbioticSphere contribute for platforms to adapt resource efficiency to dynamic demand changes while helping agents improve response time and throughput performance.

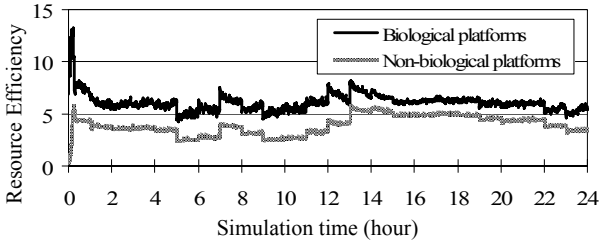

Fig. 9 Resource Efficiency

Fig. 10 shows power saving ratio compared with an always-on scenario where platforms do not have biological behaviors (replication and death). SymbioticSphere saves nearly 40% power consumption at maximum, compared with traditional grids. The average power saving ratio is 17.51%. Fig. 9 demonstrates that platforms of SymbioticSphere autonomously decrease their population when service request rate is low. When service request rate becomes higher from 12:00 to 2:00, platforms replicate themselves so that agents can handle higher workload. As a result, power saving ratio decreases to 0%. Fig. 10 and 8 show that the biological mechanisms in SymbioticSphere contribute for platforms to improve

power efficiency and adaptively trade power efficiency and agent performance (response time and throughput).

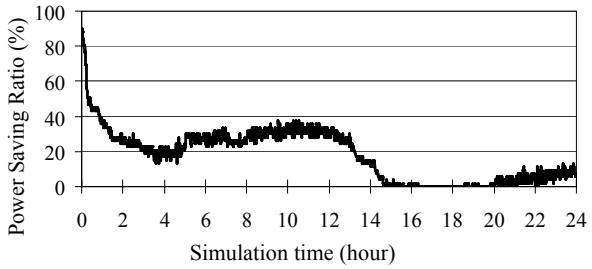

Fig. 10 Power Saving Ratio

Figs. 11, 12 and 13, show the average response time, average throughput and average power saving ratio in different network sizes. The same service request rate (shown in Fig. 5) is placed on 5x5, 7x7 and 10x10 networks. The response time and throughput improve when network size increases, because agents and platforms can utilize more resources to process service requests. The energy saving ratio increases when network size increases, because more platforms can deactivate idle hosts during low service request rate. In a 10x10 network, SymbioticSphere achieves over 50% power saving.

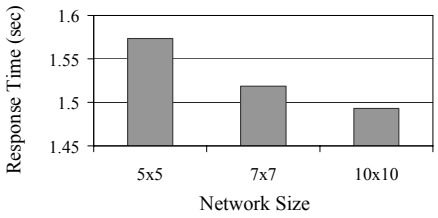

Fig. 11 Response Time in Different Network Size

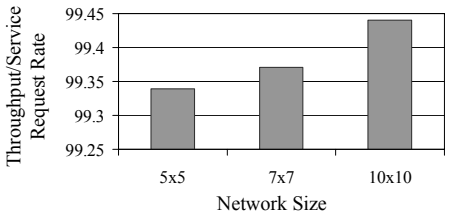

Fig. 12 Throughput in Different Network Size

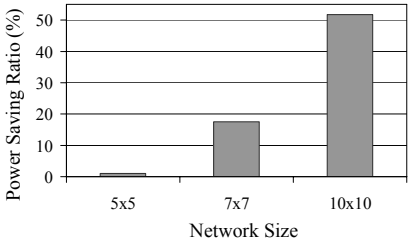

Fig. 13 Power Saving Ratio in Different Network Size

5. Related Work

This work is an extension to the Bio-Networking Architecture [11, 15]. In this architecture, biologically-inspired agents adapt their population and location to dynamic environmental changes in a decentralized manner. However, platforms do not adapt to environmental changes because they are static and non-biological entities. Also, the Bio-Networking Architecture does not consider power efficiency. In SymbioticSphere, both agents and platforms are biological entities, and they achieve autonomous adaptability and power efficiency.

In order to improve power efficiency in server farms, there are typical two approaches: (1) decreasing the frequency and voltage of CPUs (e.g., [16, 17]) and (2) turning off idle hosts (e.g., [19]). The first approach requires dynamic voltage scaling (DVS) CPUs; it cannot be used with standard CPUs. Moreover, CPU power consumption is generally small (approximately 15%) in the total power consumption in a host [18]. The power saving with this approach is limited. SymbioticSphere follows the second approach. Network hosts are turned off when platforms die on the hosts, and they are turned on when platforms replicate themselves on the hosts.

Following the above second approach, [19] addresses power efficiency of cluster systems. It is similar to SymbioticSphere in that it turns on and off hosts based on the workload placed on them. In [19], a centralized workload monitor periodically inspects resource utilization on hosts (e.g., CPU, disk and bandwidth utilization) and decides which hosts to be turned on and off. Thus, the workload monitor needs to be always on; power saving is not applied on the workload monitor. In contrast, SymbioticSphere does not rely on any centralized entities. Individual hosts make turn-on/off decisions themselves. Power saving is applied to all the hosts in the network.

Resource Broker [20] and Muse [21] are designed to dynamically adjust resource allocation 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 power consumption of a server cluster (i.e., multiple servers) and adjusts resource allocation for applications. Rather than following centralized architectures, agents and platforms in SymbioticSphere decide where to run themselves in a decentralized manner. Also, Resource Broker does not consider power efficiency of server clusters. Muse addresses power efficiency by monitoring power consumption of servers and changing resource allocation. In contrast, platforms in SymbioticSphere do not monitor power consumption of hosts. Instead, they monitor health level (i.e., resource availability) of local hosts and adjust their population. Power efficiency is a result of emergence from collective actions of platforms (i.e., replication and death behaviors).

Rainbow investigates the adaptability of server clusters [22]. A centralized system monitor periodically inspects the current environment conditions (e.g., workload placed on hosts), and performs an adaptation strategy (e.g., service migration and platform replica-

tion/removal). SymbioticSphere implements more adaptation strategies such as agent replication (service replication) and agent death (service removal). It also addresses power efficiency as well as adaptability with the same set of agent/platform behaviors. Rainbow does not consider power efficiency.

[23] proposes a decentralized design for server clusters to guarantee response time to users. 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 in SymbioticSphere can adapt more system measures to dynamic environmental changes than [23] does, such as throughput and power consumption. [23] does not consider power efficiency.

The concept of energy in SymbioticSphere is similar to money in economy. MarketNet [24] and WALRAS [25] apply the concept of money to address market-based access control for network applications. Instead of access control, SymbioticSphere focuses on adaptability and power efficiency of network systems (network applications and middleware platforms).

[26] implements the concept of symbiosis between groups of peers (hosts) in peer-to-peer networks. Peer groups symbiotically connect or disconnect with each other in order to improve the speed and quality of queries. A special type of peers implements the symbiotic behaviors for peer group connection/disconnection. Since the number of the symbiotic peers is statically fixed, they do not scale well to network size and traffic volume. They do not address power efficiency. In SymbioticSphere, all agents and platforms are designed in a symbiotic manner They scale well to network size and traffic volume, and achieve power consumption efficiency.

6. Conclusion

This paper overviews the design of SymbioticSphere, and evaluates how its biological mechanisms impact the adaptability and power efficiency of server farms. Simulation results show SymbioticSphere allows server farms to autonomously improve their adaptability and power efficiency without sacrificing their performance.

Reference

[1] IDC, "Server Power Consumption Reemerges as a Critical Cost Factor in Datacenters," Document#: 33937, August 2005.

[2] K. Kawamoto, J. Koomey, B. Nordman, R. Brown, M. Piette, M. Ting, A. Meier, "Electricity Used by Office Equipment and Network Equipment in the US," Technical Report LBNL-45917, Lawrence Berkeley National Laboratory, February 2001.

[3] J. Fuller, "U.S. Officials Cite Serious Energy Shortage," Washington File, April 2001.

[4] S. Manley, and M. Seltzer, "Web Facts and Fantasy," *Proc. of USENIX Symposium on Internet Technologies and Systems*, 1997.

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

[6] J. Challenger, P. Dantzig, and A. Iyengar, "A Scalable and Highly Available System for Serving Dynamic Data at Frequently Accessed Web Sites," *Proc. of ACM/IEEE SC'98*, November 1998.

[7] Seeley, T., *The Wisdom of the Hive*, Harvard University Press, 1995.

[8] N. Minar, K. H. Kramer and P. Maes, "Cooperating Mobile Agents for Dynamic Network Routing," *Software Agents for Future Communications Systems*, Chapter 12, Springer, 1999.

[9] G Cabri, L. Leonardi and F Zambonelli, "Mobile-Agent Coordination Models for Internet Applications," *IEEE Computer*, February 2000.

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

[11] 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.* February 2005.

[12] P. Gunaratne, K. Christensen, and B. Nordman, "Managing Energy Consumption Costs in Desktop PCs and LAN Switches with Proxying, Split TCP connections, and Scaling of Link Speed," *Int'l J. of Network Management,* Vol. 15, No. 5, 2005.

[13] Advanced Micro Devices, Inc., *Magic Packet Technology*, Technical White Paper 20213, November 1995.

[14] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. W. Keller. "Energy management for commercial servers," *IEEE Computer*, 36(12):39-48, December 2003.

[15] T. Suda, T. Itao and M. Matsuo, "The Bio-Networking Architecture: The Biologically Inspired Approach to the Design of Scalable, Adaptive, and Survivable/Available Network Applications," *The Internet as a Large-Scale Complex System*, Oxford University Press, June 2005.

[16] Sharma, A. Thomas, T. Abdelzaher, Z. Lu, and K. Skadron, "Power-Aware QoS Management on Web Servers." *Proc. of the 24th Int'l Real-Time Systems Symposium*, Dec. 2003

[17] M. Elnozahy, M. Kistler, and R. Rajamony, Energy Conservation Policies for Web Servers. *Proc. of the 4th USENIX Symposium on Internet Technologies and Systems*, March 2003

[18] Intel Corporation, *PC Energy-Efficiency Trends and Technologies*, technical white paper, 2002.

[19] E. Pinheiro, R. Bianchini, E. Carrera, and T. Heath, "Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems," Technical Report DCS-TR-440, Rutgers University, May 2001

[20] A. Othman, P. Dew, K. Djemame, I, Gourlay, "Adaptive Grid Resource Brokering," *Proc. of IEEE Int'l Conference on Cluster Computing*, Dec. 2003

[21] J. Chase, D. Anderson, P. Thakar, and A. Vahdat, "Managing Energy and Server Resources in Hosting Centers," *Proc. of ACM SOSP'01*, October 2001

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

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

[24] Y. Yemini, A. Dailianas, and D. Florissi, "MarketNet: A Market-based Architecture for Survivable Large-scale Information Systems," *Proc. of ISSAT International Conference on Reliability and Quality in Design*, Aug. 1998

[25] M. P. Wellman, "A Market-Oriented Programming Environment and Its Application to Distributed Multicommodity Flow Problems," *Journal of Artificial Intelligence Research*, Vol. 1, 1993.

[26] N. Wakamiya and M. Murata, "Toward Overlay Network Symbiosis," *Proc. of P2P 2005*, September 2005.