AUTONOMIC COMPUTING THE NEW VISION OF COMPUTING

Priyanka Parve[1]
[1]Lecturer, Department of Computer Engineering, Rajarambapu Institute of Polytechnic, Pune, India

Running Title: *Research in Computer Science and Engineering*

ABSTRACT: Various projects have been undertaken by various software companies to create the world of self-managing and intelligent computers that requires little or no human interaction. Self-management capabilities include: self-healing, dynamic workload management, self-protection. The intelligent computers would be capable of handling simple tasks, such as correcting system failures configuring themselves by installing new operating system software and data automatically performing a wider variety of tasks, while crashing less often.

Keywords: Autonomic, virtual, hermetic environment, unpredictable, status quo, akin.

INTRODUCTION:
Helping computers help themselves

If you're being chased by a big snarling dog, you don't have to worry about adjusting your heart rate or releasing a precise amount of adrenaline. Your body automatically does it all, thanks to the autonomic nervous system, the master-control for involuntary functions from breathing and blood flow to salivation and digestion.

Computers, they say, are dumb on their own-just a piece of circuitry junk. These silicon slaves are brought to life only by their master's imagination and wish. But soon the genie would be on its own-it wouldn't be dumb anymore. Computer hardware increases in speed and capacity by factors of thousands each decade; computer software piles on new features and fancier interfaces nearly as fast. So why do computers still waste our time and drive us crazy?

In Autonomic Computing system, when the behavior of an element of the computing system starts showing the first indications of distress, automatic services would fire up backup systems, order replacement parts or take other measures to ensure that people using the system don't notice problems. Autonomic Computing comprises an open standards-based system, one that: tracks its own resources, so it can share them with other systems; repairs itself; and maximizes how it uses its resources so that workloads can be moved to whichever servers and data storage facilities will process them most efficiently. The system also protects itself from viruses and hackers and anticipates users' information needs. The trick is to embed a layer of smart middleware deep in the system. This middleware will monitor system performance and execute repairs, resource allocations, and applications as necessary, without barraging network

administrators with new parameters to set and operational decisions to make.

IBM, Sun, H-P, Microsoft, and their competitors are working to automate various information technology (IT) services, notably hardware and software maintenance and support, where autonomic-like products could grab a share of a large and growing market.

Since the 1970s, scientists have researched artificial intelligence, seeking to create computers that could reason like humans, supercomputers and the business servers developed by various projects won't replicate human intelligence. Rather, they will crunch numbers at extraordinary speeds, processing trillions of transactions a second.

It's time to design and build computing systems capable of running themselves, adjusting to varying circumstances, and preparing their resources to handle most efficiently the workloads we put upon them. These autonomic systems must anticipate needs and allow users to concentrate on what they want to accomplish rather than figuring how to rig the computing systems to get them there.

AUTONOMIC COMPUTING:
THE NEXT ERA OF COMPUTING

The information technology industry loves to prove

the impossible possible. They obliterate barriers and set records with astonishing regularity. But now we face a problem springing from the very core of our success. More than any other I/T problem, this one, if it remains unsolved will actually prevent us from moving to the next era of computing. Interestingly enough, it has little to do with the usual barriers that preoccupy us. It's not about keeping pace with Gordon Moore's Law, which states that the number of transistors on a chip would grow exponentially with each passing year, but rather dealing with the consequences of its decades-long reign. It's not directly related to how many bits we can squeeze into a square inch, or how thinly we can etch lines in silicon. In fact, a continued obsession with the smaller/faster/cheaper triumvirate is really a distraction. It's not a barrier of "machine intelligence," either, that threatens our progress. It has less to do with building "thinking machines" that embody the popular conception of artificial intelligence (AI) than automating the day-to-day functioning of computing systems. We don't really need a bett hess-playing supercomputer—or sentient machine and androids programmed to love and laugh — to overcome the largest obstacle standing in our way. The obstacle is complexity. Dealing with it is the single most important challenge facing the I/T industry.

It is the next Grand Challenge.

To make individuals and businesses more productive by automating key tasks and processes and for good reason: in the evolution of humans and human society automation has always been the foundation for progress. Relegate life's mundane requirements to "automatically handled," and we free our minds and resources to concentrate on previously unattainable tasks. About how we'll connect with a friend halfway across the globe—we simply pick up the phone. But evolution via automation also produces complexity as an unavoidable byproduct. Computing systems especially have proved this true. Follow the evolution of computers from single machines to modular systems to personal computers networked with larger machines and an unmistakable pattern emerges: incredible progress in almost every aspect of computing—microprocessor power up by a factor of 10,000, storage capacity by a factor of 45,000, communication speeds by a factor of 1,000,000—but at a price. Along with that growth has come increasingly sophisticated arc hitectures governed by software whose complexity now routinely demands tens of millions of lines of code. Some operating environments weigh in at over 30 million lines of code created by over 4,000 programmers!

The Internet adds yet another layer of complexity by allowing us to connect, this world of computers and computing systems with telecommunications networks. In the process, the systems have become increasingly difficult to manage and, ultimately, to use—ask anyone who's tried to merge two I/T systems built on different platforms. In fact, the growing complexity of the I / T infrastructure threatens to undermine the very benefits information technology aims to provide. Up until now, we've relied mainly on human intervention and administration to manage this complexity.

FEATURES OF AUTONOMIC COMPUTING

1 To be autonomic, a computing system needs to "know itself"—and comprise components that also possess a system identity. Since a "system" can exist at many levels, an autonomic system will need detailed knowledge of its components, current status, ultimate capacity, and all connections with other systems to govern it. It will need to know the extent of its "owned" resources, those it can borrow or lend, and those that can be shared or should be isolated. Such system definition might seem simple, and when a "computer system" meant one room-filling machine, or even hundreds of smaller machines networked within the walls of one company, it was. But link those hundreds of computers to millions more over the Internet, make them interdependent, and allow a global audience to link back to those hundreds of computers via a proliferating selection of access devices—cell phones, T Vs, intelligent appliances— and we have blurred the once-clear concept of a "system." Start allowing all those devices to share processing cycles, storage and other resources, add to that the possibility of utility-like leasing of computing services, and we arrive at a situation that would seem to defy any definition of a single "system." But it's precisely this awareness at an overall system-wide level that autonomic computing requires. A system can't monitor what it doesn't know exists, or control specific points if its domain of control remains undefined.

To build this ability into computing systems, clearly defined policies embodied in adaptable software agents will have to govern a system's definition of itself and its interaction with I/T systems around it. These systems will also need the capacity to merge automatically with other systems to form new ones, even if only temporarily, and break apart if required into discrete systems.

2 An autonomic computing system must configure and reconfigure itself under varying and unpredictable conditions. System configuration or "setup" must occur automatically, as must dynamic adjustments to that configuration to best handle changing environments. Given possible permutations in complex systems, configuration can be difficult and time-consuming — some servers alone present hundreds of configuration alternatives. Human system administrators will never be able to perform dynamic reconfiguration as there are too many variables to monitor and adjust—multiply hundreds of alternatives by thousands of servers and other system devices—in too short a period of time—often minutes, if not seconds. To enable this automatic configuration ability, a system may need to create multiple images of critical software, such as an operating system (a kind of software cloning), and reallocate its resources (such as memory, storage, communications bandwidth, and processing) as needed. If it is a globally distributed system, it will need to leverage its multiple images and backup copies to recover from failures in localized parts of its network. Adaptive

algorithms running on such systems could learn the best configurations to achieve mandated performance levels.

3 An autonomic computing system never settles for the status quo , it always looks for ways to optimize its workings. It will monitor its constituent parts and fine-tune workflow to achieve predetermined system goals, much as a conductor listens to an orchestra and adjusts its dynamic and expressive characteristics to achieve a particular musical interpretation. This consistent effort to optimize itself is the only way a computing system will be able to meet the complex and often conflicting I/T demands of a business, its customers, suppliers and employees. And since the priorities that drive those demands change constantly, only constant self-optimization will satisfy them.

We'll need to answer questions such as how often a system takes control actions, how much delay it can accept between an action and its effect, and how all this affects overall system stability. Innovations in applying control theory to computing must occur in tandem with new approaches to overall systems architecture, yielding systems designed with control objectives in mind . Algorithms seeking to make control decisions must have access to internal metrics. And like the tuning knobs on a radio, control points must affect the source of those internal metrics. Most important, all the components of an autonomic system, no matter how diverse, must be controllable in a unified manner.

4 An autonomic computing system must perform something akin to healing — it must be able to recover from routine and extraordinary events that might cause some of its parts to malfunction. It must be able to discover problems or potential problems, then find an alternate way of using resources or reconfiguring the system to keep functioning smoothly. Instead of "growing" replenishment parts, as our cells do, healing in a computing system means calling into action redundant or underutilized elements to act as replacement parts. (In a sense, this is akin to what our brain does when parts of it are damaged.) Of course, certain types of "healing" have been a part of computing for some time. Error checking and correction, an over 50-year-old technology, enables transmission of data over the Internet to remain remarkably reliable, and redundant storage systems like RAID allow data to be recovered even when parts of the storage system fail.

But the growing complexity of today's I/T environment makes it more and more difficult to locate the actual cause of a breakdown, even in relatively simple environments. In more complex systems, identifying the causes of failures calls for root-cause analysis (an attempt to systematically examine what did what to whom and to home in on the origin of the problem). But since restoring service to the customer and minimizing interruptions is the primary concern, an action-oriented approach (determining what immediate actions need to be taken given current information available) will need to take precedence in an autonomic solution. Initially, "healing" responses taken by an autonomic system will follow rules generated by human experts. But as we embed more intelligence in computing systems, they will begin to discover new rules on their own that help them use system redundancy or additional resources to recover and achieve the primary objective: meeting the goals specified by the user.

5 A virtual world is no less dangerous than the physical one, so an autonomic computing system must be an expert in self-protection. It must detect, identify and protect itself against various types of attacks to maintain overall system security and integrity. Before the Internet, computers operated as islands. It was fairly easy then to protect computer systems from attacks that became known as "viruses." As the floppy disks used to share programs and files needed to be physically mailed or brought to other users, it took weeks or months for a virus to spread. The connectivity of the networked world changed all that. Attacks can now come from anywhere. And viruses spread quickly— in seconds— and widely, since they're designed to be sent automatically to other users. The potential damage to a company's data, image and bottom line is enormous. More than simply responding to component failure, or running periodic checks for symptoms, an autonomic system will need to remain on alert, anticipate threats, and take necessary action. Such responses need to address two types of attacks: viruses and system intrusions by hackers.

By mimicking the human immune system, a "digital immune system "—an approach that exists today—can detect suspicious code, automatically send it to a central analysis center, and distribute a cure to the computer system. The whole process takes place without the user being aware such protection is in process. To deal with malicious attacks by hackers, intrusion systems must automatically detect and alert system administrators to the attacks. Currently, computer security experts must then examine the problem, analyze it and repair the system. As the scale of computer networks and systems keeps expanding and the likelihood of hacker attacks increases, we will need to automate the process even further. There won't be enough experts to handle each incident.

6 An autonomic computing system knows its environment and the context surrounding its activity, and acts accordingly. This is almost self-optimization turned outward: an autonomic system will find and generate rules for how best to interact with neighboring systems. It will tap available resources, even negotiate the use by other systems of its underutilized elements, changing both itself and its environment in the process—in a word, adapting. This context-sensitivity includes improving service based on knowledge about the context of a transaction. Such an ability will enable autonomic systems to maintain reliability under a wide range of anticipated circumstances and combinations of circumstances. But more significantly, it will enable them to provide useful information instead of confusing data. Autonomic systems will need to be able to describe themselves and their available resources to other systems, and

they will also need to be able to automatically discover other devices in the environment. Current efforts to share supercomputer resources via a "grid" that connects them will undoubtedly contribute technologies needed for this environment-aware ability. Advances will also be needed to make systems aware of a user's actions, along with algorithms that allow a system to determine the best response in a given context.

7 An autonomic computing system cannot exist in a hermetic environment. While independent in its ability to manage itself, an autonomic computing system must function in a heterogeneous world and implement open standard s—in other words, an autonomic computing system cannot, by definition, be a proprietary solution. In nature, all sorts of organisms must coexist and depend upon one another for survival (and such biodiversity actually helps stabilize the ecosystem). In today's rapidly evolving computing environment, an analogous coexistence and interdependence is unavoidable. Businesses connect to suppliers, customers and partners.

Current collaborations in computer science to create additional open standards have allowed new types of sharing: innovations such as Linux, an open operating system; Apache, an open web server; UDDI, a standard way for businesses to describe themselves, discover other businesses and integrate with them; and from the Globus project , a set of protocols to allow computer resources to be shared in a distributed (or "grid-like") manner. These community efforts have accelerated the move toward open standards, which allow for the development of tools, libraries, device drivers, middleware, applications, etc., for these platforms. Advances in autonomic computing systems will need a foundation of such open standards.

8 Perhaps most critical for the user, an autonomic computing system will anticipate the optimized resources needed while keeping its complexity hidden. This is the ultimate goal of autonomic computing: the marshaling of I/T resources to shrink the gap between the business or personal goals of our customers, and the I / T implementation necessary to achieve those goals—without involving the user in that implementation. Even custom-made solutions rarely interact seamlessly with all a company's other I/T systems, let alone all its data and documents.

While some aspects of computing have improved for general users—graphical interfaces, for instance, are far easier for most people to use than command prompts and their corresponding dictionaries of commands—tapping the full potential of entire I/T systems is still too difficult. When faced with a potentially dangerous or urgent situation, our autonomic nervous system anticipates the potential danger before we become aware of it. It then "optimizes" our bodies for a selection of appropriate responses — specifically, the autonomic nervous system triggers our adrenal glands to flood the body with adrenaline, a hormone that supercharges the ability of our muscles to contract, increases our heart rate and breathing, and generally constricts blood vessels to increase our blood pressure (while dilating those that feed key areas such as the skeletal muscles). The net result: our body is superbly prepped for action, but our conscious mind remains unaware of anything but the key pieces of information required to decide whether to stay and act (the "fight" response) or run for the hills.

An autonomic system will allow for that kind of anticipation and support. It will deliver essential information with a system optimized and ready to implement the decisions users make and not needlessly entangle them in coaxing results from the system.

FUTURE SCOPE OF AUTONOMIC COMPUTING

Realistically, such systems will be very difficult to build and will require significant exploration of new technologies and innovations. That's why we view this as a Grand Challenge for the entire I / T industry. We'll need to make progress along two tracks: 1)making individual system components autonomic

2)achieving autonomic behavior at the level of global enterprise i/t systems.

That second track may prove to be extremely challenging. Unless each component in a system can share information with every other part and contribute to some overall system awareness and regulation , the goal of autonomic computing will not really be reached. So one huge technical challenge entails figuring how to create this "global" system awareness and management. It's not something we currently know how to do. We know there are also many interim challenges: how to create the proper "adaptive algorithms", sets of rules that can take previous system experience and use that information to improve the rules.

How to balance what these algorithms "remember" with what they ignore. We humans tend to be very good at the latter—we call it "forgetting" and at times it can be a good thing: we can retain only significant information and not be distracted by extraneous data. Still another problem to solve: how to design an architecture for autonomic systems that provides consistent interfaces and points of control while allowing for a heterogeneous environment. We could go on, as the list of problems is actually quite long, but it is not so daunting as to render autonomic computing another dream of science fiction.

First, many established fields of scientific study can contribute to autonomic computing. What we've learned in artificial intelligence, control theory, complex adaptive systems and catastrophe theory, as well as some of the early work done in cybernetics, will give us a variety of approaches to explore.

Current research projects at laboratories and universities include self-evolving systems that can monitor themselves and adjust to some changes, "cellular" chips

capable of recovering from failures to keep long-term applications running, heterogeneous workload management that can balance and adjust workloads of many applications over various servers, and traditional control theory applied to the realm of computer science, to name just a few. Also, some aspects of autonomic computing are not entirely new to the I/T industry.

It's a vision that requires the involvement of the top minds in the technology community. This next era of computing will enable progress and abilities we can barely envision today. But the best measure of our success will be when our customers think about the functioning of computing systems about as often as they think about the beating of their hearts.

IBM Corp. has taken the wraps off a multibillion-dollar research initiative to bring some of the self-healing abilities of living creatures to the brittle world of computers. Projects eLiza-one of the most ambitious undertakings by IBM-is the company's answer to the rapidly growing challenge of managing the increasingly complex computer systems necessary for doing business on the Internet.

PROJECT eLIZA

There has been a relentless proliferation of faster, more powerful technology. Exponential improvements in price/performance have become almost commonplace, but IBM Research studies indicate price /performance, in a number of cases, is actually super-exponential, that is the rate of improvement is actually going up.

Emerging from the profusion of technology is a complex, multifaceted infrastructure, one both distributed and centralized; used by small and large business alike; focussed on content and transactions; supporting old and new applications; partially owned and partially outsourced.

Clearly, such an environment poses major challenges to our industry: among them, how to manage such a far-flung, complex, and profoundly indispensable infrastructure, specially, how to take advantage of the latest management technology, while keeping costs under control. The world faced a similar challenge in the 20th century when the telephone began to proliferate. Initially, telephone calls were switched manually by human operators, a perfectly good solution when Alexander Graham Bell and his man Thomas Watson were the only ones with phones.

But had the same management model been followed as the telephone's popularity grew, Ma Bell before long would have had to employ every man, woman and child on earth to manage the volume of calls. The solution was automation; the infrastructure had to mange itself.IBM's Project eLiza is about using technology to manage technology. Project eLiza is an ongoing effort to create servers that respond to unexpected capacity demands and system glitches without human intervention.

The ultimate goal of the IBM project is to create "intelligent" computers capable of handling simple tasks by themselves, such as correcting system failures and warding off attacks from hackers.

The goal: new highs in reliability, availability and serviceability, and new lows in downtime and cost of ownership. Project eLiza has made self-management capabilities possible throughout IBM families. Traits shared by xSeries, iSeries, zSeries and pSeries servers include:

- Support for dynamic clustering.
- Support for dynamic partitioning.
- EZ Set Up wizards, allowing for self-installation.

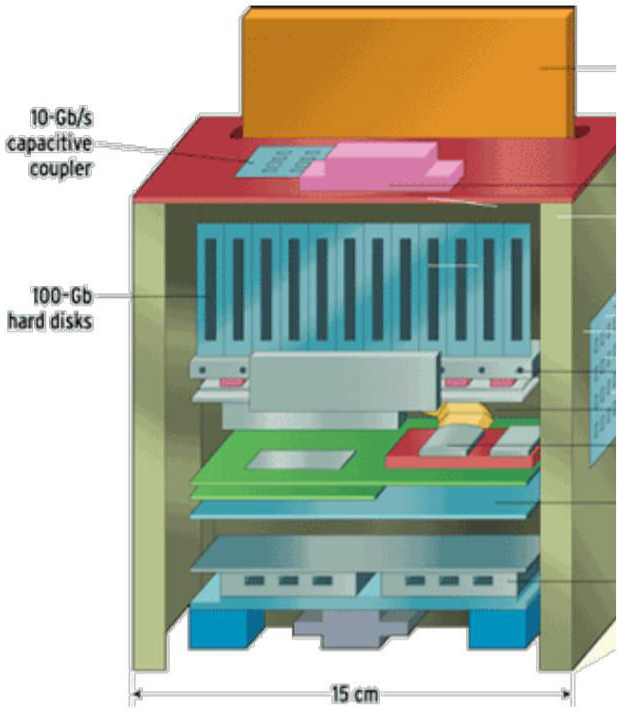
User authentication, directory integration and other tools to protect access to network resources. Heterogeneous enterprise-wide workload management. Larger and larger systems, networks of networks, billions of users, a trillion devices, colossal volumes of transactions: all mean new levels of complexity, more and more pieces that must be configured and tuned, more and more points at which systems can fail or intruders penetrate.

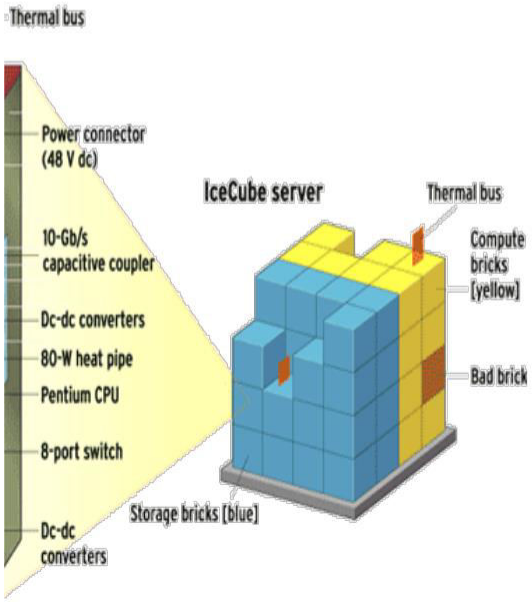
BUILDING STORAGE:
BRICK BY INTERCHANGEABLE BRICK:

IBM Research has already started to guide the company down the autonomic path.Offerings in its family of relational databases called Database 2 (DB2) maintain themselves and optimize querying. Servers that automatically allocate storage and computing resources are moving to market, thanks to the Server Group's Project eLiza.

Still in prototype is a redundant server system based on storage and compute "bricks" called the IceCube server. Comprising an array of Collective Intelligent Bricks (CIBs), IceCube is the smarter son of industry-standard Redundant Array of Inexpensive Disks (RAID) systems. The IceCube server tracks its own components, health, maximum capacity, and relationship to other systems. It can distribute data among many CIBs even though your computer thinks they're all in one place. The system works around any failed node, negating the urgency, or even the need, for repair. Failed bricks are left in place until there comes a convenient time to fish out the dead soldier

IBM is building a self-managing, modular data storage server, known as IceCube, using Collective Intelligent Bricks (CIBs). Each CIB, 15 cm on a side, is outfitted with twelve 100-GB hard disks, a Pentium microprocessor running Linux, and six 10-Gb/s capacitive couplers to interface each face of the brick with the adjacent one. A prototype containing 27 bricks will be completed by year's end.





CLUSTERING

Clustering can be best described as a technology that automatically allows one physical server to take over the tasks and responsibility of another physical server that has failed. The obvious goal behind this, given that all computer hardware and software will eventually fail, is to ensure that user running mission-critical application will have or no downtime when such a failure occurs.

More specifically, clustering refers to a group of two or more servers (generally called nodes) that work together and represent themselves as a single virtual server to a network. In other words, when a client connects to clustered Servers, it thinks there is only a single Server, not more than one. When one of the nodes fails, its responsibilities are taken over by another server in the cluster, and the end-user notice little, if any differences before, during, and after the failover.

➢ **WORKING OF CLUSTERRED COMPUTER:**

In a two-cluster node, one of the Servers is referred to as the primary node, and the second one is referred to as the secondary node. In an Active/Passive cluster design, Server will run on the primary node, and should the primary node fail, then the secondary node will take over.Besides both servers being connected to shared disk array, both nodes of the clusters are also connected to each other via a private network. This private network is used for each node to keep track of the status of the other node. Essentially what happens in a Server Cluster is that you assign Server its own virtual name and virtual TCP/IP address. This name and address is shared by both of the servers in the cluster.

Typically, a client will connect to the Server cluster using the virtual name used by the cluster. And as far as a client is concerned, there is only one physical Server, not two. Assuming that the primary node of the Server cluster is the node running Server on an Active/Passive cluster design, then the primary node will respond to the client's requests. But if the primary node fails, and failover to the secondary node occurs, the cluster will still retain the same Server virtual name and TCP/IP address, although now a new physical server will be responding to client's requests.

During the failover period, which can last several minutes clients will be unable to access Server, so there is a small amount of downtime when failover occurs. Some software will just wait the failover out, and when the failover completed, it will continue just as nothing had happened. Some software will present a message box on the screen, describing a lost connection. Other client software will not know what to do, and users may have to exit, and then reload the client before they can access Server again.

CONCLUSION:

For the past 30 years, the mantra of the field was faster, bigger and cheaper. Slowly, and finally, the goal may be changing to easier, more reliable and less deserving of a swift kick. For the past years we have focused on performance. We built things that went really fast, but they didn't work.

In the process, computing systems have become increasingly difficult to manage. In fact, this growing complexity of the IT infrastructure threatens to undermine the very benefits information technology aims to provide. Up until now, we've relied mainly on human administration to manage this complexity.

Like Charlie Chaplin falling into the machine in "Modern Times," the people responsible for the smooth running of these interconnected systems are starting to get in

the way. Ironically, it will take higher levels of complexity to solve this problem. Future computer designs will shift away from today's race for performance to concentrate on efforts to craft self-managing systems. Those systems will mask their growing complexity and interoperate with one another in useful, automated ways.

Many researchers are pursuing the concept of grid computing as the next computing model beyond today's client/server systems. In this setup, systems in one data center might reach out to others across town or across the globe for help during peak hours Whether it's called autonomic computing, grid computing or Web services, an expanding pool of researchers see computer design taking shape around concepts in distributed computing that look more at automating the interplay between complex systems and less at bolstering the performance of an individual system or CPU.

Future computing is a vision to design computers that can configure, maintain and repair themselves without human intervention

REFERENCES:

[1] Manish Paraskar, Salim Hariri, "Autonomic computing, An Overview", January 2004.

[2] JAyapalan anitha, Raja, Selvaraju, "Autonomic computing", 2011.

[3] Glen fink, Deb Frink, "Autonomic Computing, Freedom or Threat", April 2007

[4] Jeffrey O. Kephart David M. Chess IBM Thomas J. Watson Research Center, "Vision of Autonomic computing, 2003.

[5] D. M. Chess, C. Palmer, and S. R. White. Security in an autonomic computing environment. In IBM System Journal, volume 42, pages 107– 118, January 2003.

[6] A. G. Ganek and T. A. Corbi. The dawning of the autonomic computing era. In IBM System Journal, volume 42, pages 5–18, January 2003.

[7] IBM Corporation Software Group. The Tivoli software implementation of autonomic computing guidelines. In Available at http://www03.ibm.com/autonomic/pdfs/br-autonomic-guide.pdf, 2002.

[8] S. Hariri and M. Parashar. Autonomic Computing: An overview. In Springer-Verlag Berlin Heidelberg, pages 247–259, July 2005.

[9] J. O. Kephart. Research challenges of autonomic computing. In Proceedings of the 27th International Conference on Software Engineering, pages 15–22, May 2005.

[10] J. O. Kephart and D. M. Chess. The vision of autonomic computing. In IEEE Computer, volume 36, pages 41–50, January 2003.

[11] P. Leaney, A. MacArthur, and J. Leaney. Defining Autonomic computing: A software engineering perspective. In Australian Software Engineering Conference (ASWEC'05), 2005.

[12] J. A. McCann and M. C. Huebscher. Evaluation issues in autonomic computing. In Proceedings of Grid and Cooperative Computing workshop(GCC), volume 15, pages 597–608, October 2004.

[13] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-Peer Computing. In Proceedings of the Second International Conference on Peer-to-Peer Computing, pages 1–51, July 2002.

[14] R. Murch. Autonomic Computing. In Prentice-Hall, pages 0–20:25–40, October 2004.

[15] M. Parashar, Z. Li, H. Liu, V. Matossian, and C. Schmidt. Enabling Autonomic Grid Applications: Requirements, Models and Infrastructures. In Self-Star Properties in Complex Information Systems, Lecture Notes in Computer Science, Springer Verlag, volume 3460, 2005.

[16] M. Salehie and L. Tahvildari. Autonomic Computing: emerging trends and open problems. In ACM SIGSOFT Software Engineering Notes, volume 30, pages 1–7, July 2005.

[17] R. Sterritt and D. Bustard. Towards an autonomic computing environment. In 14th International Workshop on Database and Expert Systems Applications (DEXA '03), pages 694–698, September 2003.

[18] R. Sterritt, M. Parashar, H. Tianfield, and R. Unland. A concise introduction to autonomic computing. In Advanced Engineering Informatics, volume 19, pages 181–187, January 2005. [15] G. Tesauro and et al. A Multi-agent systems approach to autonomic computing. In IBM Press, pages 464–471, March 2004.

[19] H. Tianfield. Multi-agent autonomic architecture and its application in e-medicine. In IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003), pages 601–604, October 2003.

[20] S. White and et al. An architectural approach to autonomic computing. In Proceedings International Conference on Autonomic Computing (ICAC'04), NewYork, USA, pages 2–9, May 2004.

[21] T. De Wolf and T. Holvoet. Evaluation and comparison of decentralised autonomic computing systems. In Department of Computer Science, K.U.Leuven, Report CW 437, Leuven, Belgium, March 2006.