

# The Curse of Voodoo Engineering

<September 29, 2004>

---

---

# Contents

<b><u>1Voodoo Engineering Defined.....</u></b>	<b><u>3</u></b>
<b><u>2The Cause of Voodoo Engineering.....</u></b>	<b><u>4</u></b>
2.1No representative data on how our standard platforms perform.....	4
2.2No real data from vendors.....	4
<b><u>3The Cost of Voodoo Engineering.....</u></b>	<b><u>5</u></b>
<b><u>4The Cure of Voodoo Engineering.....</u></b>	<b><u>6</u></b>
4.1Standard Test Suite Usage.....	6
4.2Vendor involvement.....	6
4.3Closed feedback loop.....	6
<b><u>5Next Steps.....</u></b>	<b><u>7</u></b>

---

# 1 Voodoo Engineering Defined

The Jargon File (<http://www.catb.org/~esr/jargon/html>) defines the term "Voodoo Programming" (<http://www.catb.org/~esr/jargon/html/V/voodoo-programming.html>):

voodoo programming: n.

[from George Bush Sr.'s "voodoo economics"]

1. The use by guess or cookbook of an obscure or hairy system, feature, or algorithm that one does not truly understand. The implication is that the technique may not work, and if it doesn't, one will never know why. Almost synonymous with black magic, except that black magic typically isn't documented and nobody understands it. Compare magic, deep magic, heavy wizardry, rain dance, cargo cult programming, wave a dead chicken, SCSI voodoo.
2. Things programmers do that they know shouldn't work but they try anyway, and which sometimes actually work, such as recompiling everything.

I propose a new term "Voodoo Engineering" with the following definition:

voodoo engineering: n

1. The sizing of servers and architecture of interrelated systems by guess, vendor literature or "feeling" without reference to actual testing metrics or performance numbers. Requires the blind acceptance of whatever the vendor tells you will be required and the complete ignoring of all attempts at measuring the performance of the current systems or systems representative of the ones being designed.

Voodoo Engineering (VE) describes the usual way systems are selected and sized within most companies.

---

## 2 The Cause of Voodoo Engineering

There is a two-fold cause of VE.

### 2.1 No representative data on how standard platforms perform.

The engineering group(s) does not have the time or resources to properly evaluate the performance of the various hardware and software combinations that are currently presented as 'standards' to the LOB teams.

We do not know how any system will perform under any given load. We all have experiences that guide us, but we can not tell you that an HP DL380 running Windows 2000 Advanced Server is faster or slower, cheaper or more expensive at doing anything than an IBM P650 running AIX 5 or any other OS/hardware combination. We think we know, but we don't.

This issue will only become worse as Linux adoption increases and vendors (Sun and IBM in particular) attempt to position their hardware to take advantage of (or blunt the adoption of) Linux. For instance: With the introduction of IBM's OpenPower, Linux only line, what types of loads should be moved from AIX/pSeries to Linux/OpenPower? Ditto for Sun's introduction of their Opteron servers and new interest and support for Solaris x86. What runs best on a Sun Opteron box? Solaris? Linux? Does Linux run better or worse on a Sun Opteron as on an HP one? And for everybody's favorite...when is it appropriate and cost effective to move a load from anything to zLinux running on the mainframe? Or the other way?

### 2.2 No real data from vendors.

Vendors (and I include internal LOB development teams) do not (and perhaps can not) provide meaningful technical requirements for their systems. They do not come to us and say "This system will generate x number of page requests of y average size every minute and the response time has to be z.". Instead we get requests for "4 xyz boxes" without any demonstration of why that quantity and model are needed.

Their inability to provide meaningful technical specifications indicates a lack of understanding of their own systems at a technical level. This means that they cannot predict how they will behave in a particular environment with a particular set of hardware.

---

### 3 The Cost of Voodoo Engineering

Not surprisingly the cost of VE is found in overspending on hardware. This impacts other areas due to the costs of supporting too many servers, requiring more licenses for dozens of pieces of software, filling up data centers with idle servers, etc, but the root is in the size and number of servers.

In any project situation where an engineer or architect is asked to specify the number and size of servers, he is constrained by three factors, (which he is probably not even cognitively aware of) in any attempt he may make to influence the number or size of the servers:

- 1) the facts given under the cause of VE (ie, he has no data).
- 2) the realization that it is easier to get \$2 million now, than to get \$1 million now and an 'emergency' purchase of \$250,000 six weeks before go live when the test system bogs down. (Even though this would be a net save of \$750,000, it would be judged a failure because he had to go back for more money.)
- 3) the realization that no one is tracking it anyway. While our performance agents collect gigabytes of data, none of that is being turned into information that will be used to determine if the system is correctly sized and to reward the engineer for doing a good job (or penalize him for doing a poor one). All that matters is getting the job done 'on time' and one of the best ways to ensure that is to make sure you have plenty of hardware.

---

## 4 The Cure of Voodoo Engineering

So what is the stake that can be driven through the heart of VE? What is the silver bullet? In short "a standard, portable, test suite yielding comparative data". The engineering groups must develop a process that uses a suite of test programs. Processes and test programs that are developed or selected should be:

- 1) Representative: They should test the types of loads expected on systems within the company (ie, basic web, application load, database load, etc).
- 2) Portable: They should work on as many of the standard operating system platforms as possible with little or no change.
- 3) Atomic: It should be possible to quickly and easily install and execute the tests on a system. This will enable them to be used on a wider variety of systems.
- 4) Understandable: The engineering groups must have a thorough understanding of what the tests do and do not demonstrate and how to extrapolate from those data points.

### 4.1 Standard Test Suite Usage

The test suite must be run on every standard operating system/hardware combination. The data can then be aggregated and combined with other operating and purchasing costs to develop tools to provide guidance in the selection and sizing of systems. The aggregation of the data and purchasing costs will be done in such a way that it will be possible to compare the cost of certain core operations (disk i/o, web page transfers, compute cycles, etc) between different platforms. The establishment of cost baselines for the platforms will enable the comparison of varying application frameworks based on the cost of purchasing and maintaining the computing resources required.

### 4.2 Vendor involvement

Vendors must provide both meaningful data and systems to test. There should probably be a cutoff point so that only systems that can reasonably benefit will be subjected to testing.

### 4.3 Closed feedback loop

Data must be collected on the systems as they are implemented and fed back into the test suite to tune it for the company's environment. This will lead to increased understanding of the computing environment and confidence in the test scenarios and results.

---

## 5 Next Steps

The SIS group should be tasked with coordinating the development of the Standard Test Suite. Initially the STS can be based on Open Source tools as they are

- 1) Readily available
- 2) Cross platform
- 3) Well understood

Once a core suite is in place, application oriented groups can be leveraged to provide suitable test suites for the major software packages such as WebSphere, DB2, Oracle, etc.

Education for the engineers and LOB technical contacts will need to be provided.

The use of the STS will need to be championed at the executive level.