

## Making Your z/VM® Operating System Self-Aware

*Tim Greer*  
z/VM® System Test  
[timgreer@us.ibm.com](mailto:timgreer@us.ibm.com)



- Last updated 2018 MAR 26 by Tim Greer.

### Abstract:

The z/VM® operating system's hypervising prowess makes it an ideal platform for self-awareness.

We describe setting up a model of a running system, running as a guest on that system. With such a model, it is possible to allow the operating system to consider in advance the effects of certain actions, or choose to back out of situations apparently developing from recent actions. This is more the awareness level of "Ouch!" than "Cogito ergo sum!", but it is something you can do on your zVM® system today.

## Trademarks

The following are trademarks of the International Business Machines Corporation in the United States, other countries, or both.

Not all common law marks used by IBM are listed on this page. Failure of a mark to appear does not mean that IBM does not use the mark nor does it mean that the product is not actively marketed or is not significant within its relevant market.

Those trademarks followed by ® are registered trademarks of IBM in the United States; all others are trademarks or common law marks of IBM in the United States.

For a complete list of IBM Trademarks, see [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml):

\* IBM Systems, IBM System z10®, IBM System Storage®, IBM System Storage DS®, IBM BladeCenter®, IBM System z®, IBM System p®, IBM System i®, IBM System x®, IBM IntelliStation®, IBM Power Architecture®, IBM SureOne®, IBM Power Systems™, POWER®, POWER®, POWER7®, POWER8®, Power®, IBM z/OS®, IBM AIX®, IBM i, IBM z/VSE®, IBM z/VM®, IBM i5/OS®, IBM zEnterprise®, Smarter Planet™, Storwize®, XIV®, PureSystems™, PureFlex™, PureApplication™, IBM Flex System™, Smarter Storage

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

\* All other products may be trademarks or registered trademarks of their respective companies.

### Notes

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved.

Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

## Disclaimer

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "AS IS" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used instead.

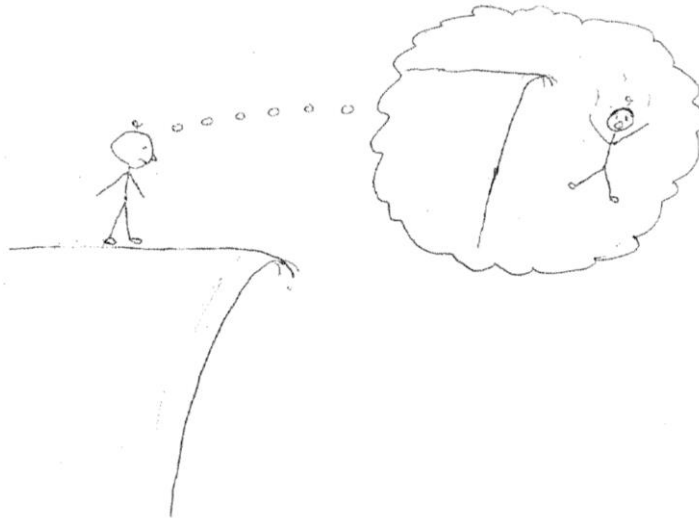
Any performance data contained in this document was determined in a controlled environment and, therefore, the results which may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environments.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming or services in your country.

## Agenda

- Motivational Problem
- What constitutes self-awareness?
- Technical design of the self-model
  - VDISKs are great!
  - 4 UserIDs
  - Initialization
  - Periodic Update
- Implementation difficulties
- Capabilities

## Basic Idea



The basic idea is to give the operating system the ability to picture itself in the future, and in particular after some proposed action. The decision on whether to take that action might then be better informed.

**Motivational Problem**

SHUTDOWN REIPL with new CPLOAD.

*What if the CPLOAD is bad?*

What does “bad” look like?

Generalization: What if I make change X to system?

## What constitutes self-awareness?

- Spectrum of awareness:  
Human Horse Fly Worm Sunflower Doorknob
- Self-awareness for a machine may look different
- “Cogito ergo sum”? ... How about, “Ouch!”
- Internal self-model. “What happens if I do this?”

It's hard to say what qualifies as self-aware. But it's a mistake to demand human-level awareness. Let's start small. I suggest a key characteristic is an internal self-model, the means to examine “What happens if I do this?”

What about a self-model of an SSI? Analogous to self-awareness of a beehive.

Rene Descartes walks into a bar...

## Model: Goals

- Self-model that the system can use for advance checking/speculation
- Everything automated – no humans required
- Low impact, small footprint
- Broadly applicable and portable
- Useful



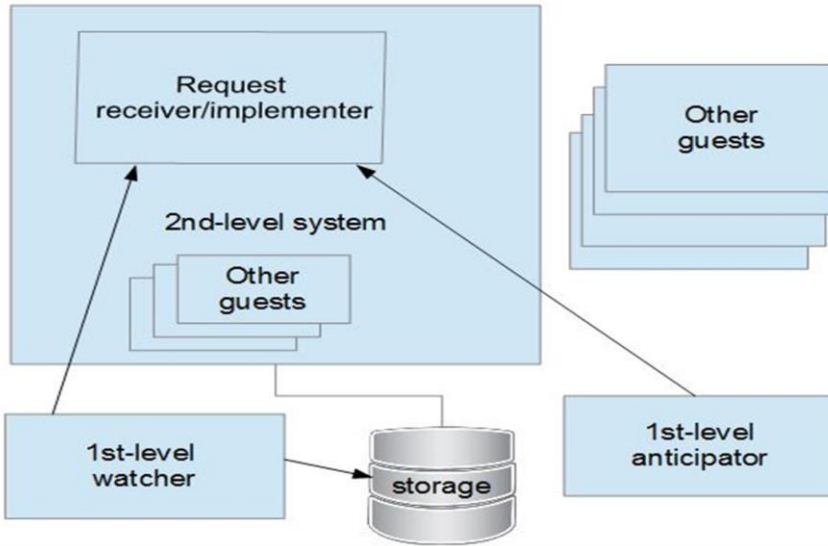
## Model

- VDISK-based 2<sup>nd</sup>-level system
- Why VDISK?
  - Always available\* on any system
  - Fast
  - Temporary – no cleanup worries
  - Temporary – no messing with DASD allocation
  - Same code can be reused for 3<sup>rd</sup>-level and up

\*If SYSTEM CONFIG allows it. See FEATURES statement.

Being able to use the same code to bring up 3<sup>rd</sup>-level makes it easy to include in our model the aspect that the system is hosting a model of itself.

### Functional Design



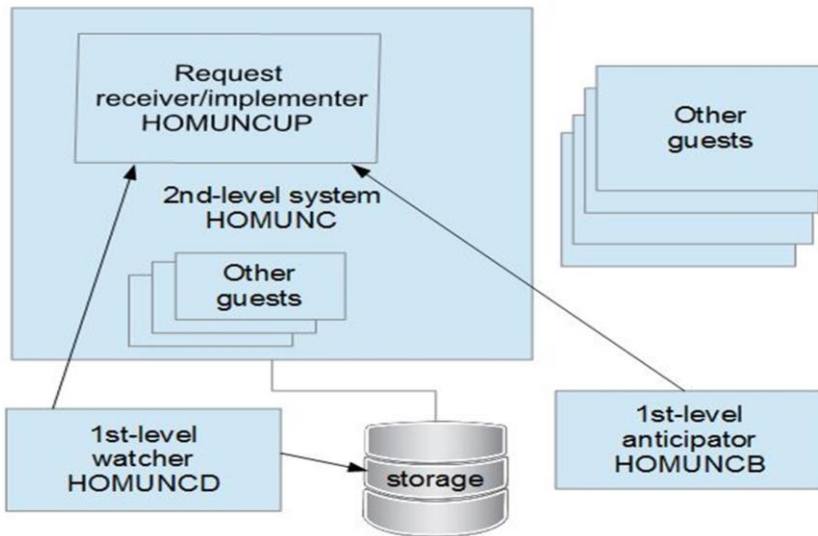
## UserIDs involved

- HOMUNC – 1<sup>st</sup>-level host of 2<sup>nd</sup>-level system
- HOMUNCD – 1<sup>st</sup>-level watcher
- HOMUNCB – 1<sup>st</sup>-level speculation conduit
- HOMUNCUP – 2<sup>nd</sup>-level servant

HOMUNCD is the only 1<sup>st</sup>-level user requiring privileges (for QUERYs and IND USER xxxx), except I also gave HOMUNCB class B for SET VARIABLE.

HOMUNC is allowed read access to 1<sup>st</sup>-level MAINT CF1 and PMAINT CF0, so that it can see the current CPLOAD MODULE and SYSTEM CONFIG, as well as CP exits and proposed new CPLOADs.

### Functional Design



Same as previous slide, but with UserIDs added.

## Initialization

- Prototype directory and SYSTEM CONFIG are predefined.
- HOMUNCD builds a file for HOMUNC to read
  - Q PROC --> define processors
  - Q CPOWN --> SYSTEM CONFIG entries
  - Q ALLOC --> SYSTEM CONFIG entries
  - Q CPLOAD
  - Q NAMES
- HOMUNC
  - Copies 1<sup>st</sup>-level SYSTEM CONFIG (mostly)
  - Adds names to prototype directory
  - Copies 1<sup>st</sup>-level CPLOAD
  - IPLs

## SYSTEM CONFIG processing

### ▪ Simplify format

- Imbed IMBEDs
- Deal with BEGIN/END, system-specific lines (with colons, e.g. GDLMSTR1: CP\_Owned Slot 1 SR1RES
- Continued lines

### ▪ Eliminate

- SSI
- SYSTEM\_IDENTIFIER\_DEFAULT
- SYSTEM\_IDENTIFIER
- SYSTEM\_RESIDENCE
- CP\_ACCESS
- CP\_OWNED
- USER\_VOLUME\_LIST
- DEVICES
- RDEVICE
- OPERATOR\_CONSOLES
- EMERGENCY\_MESSAGE\_CONSOLES

### ▪ Add

- CP\_OWNED statements based on HOMUNCD's Q CPOWN and Q ALLOC

We delete CP\_OWNED lines and then add them back in because there might have been dynamic changes.

### Periodic Update

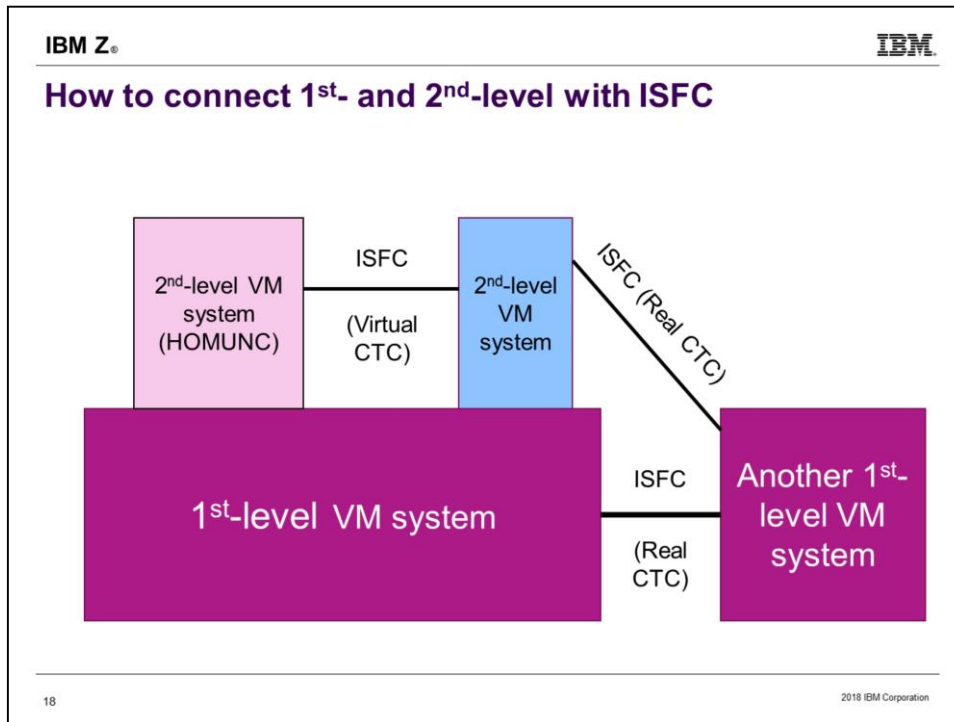
- HOMUNCD issues Q NAMES, IND USER xxxx
- If anything changed, it gives the new list to HOMUNCUP (remember HOMUNCUP is on the 2<sup>nd</sup>-level system)
- HOMUNCUP FORCES or AUTOLOGs as needed
- HOMUNCUP may have to add UserIDs to directory
- Added UserIDs are fake, but can be made to run some representative workload.

IND USER xxxx to get storage size. Could do Q PRIVCLAS too.

## Implementation difficulties -- Communications

- Requirement to communicate between 1<sup>st</sup>-level (HOMUNCB and HOMUNCD) and 2<sup>nd</sup>-level (HOMUNCUP)
- 3 alternatives considered
  - TCPIP requires configuration/connection to 1<sup>st</sup>-level stack
  - APPC requires ISFC or TSAF connection to 1<sup>st</sup>-level
  - Kludge with environment variables and shared VDISKS is slow





When configuring the hardware, if you have two or more CTCs between a pair of 1<sup>st</sup>-level systems, then you can ATTach one end of one to a (blue) 2<sup>nd</sup>-level system, which can then be a hub to which you can link any number of additional 2<sup>nd</sup>-level systems via VCTC.

TSAF can link 1<sup>st</sup>- and 2<sup>nd</sup>- level using only VCTC, no hardware. But TSAF is slow, and it will slow your ISFC collection down too.

### Implementation difficulties – Intercepting commands

- Reserve a set of commands to one UserID via privilege class
- Use EXECs to intercept CP commands
- Funnel commands through a server
- No great ideas on this yet

These are proposed methods of intercepting 1<sup>st</sup>-level commands, so we can invoke them first in the model and watch what happens. While the suggestions here will certainly work, it would be preferable to be able to issue system commands from any authorized user, like today. A CP mod, or at least a CP exit, would seem to be required.

## Capabilities

- What if...?
- Test effects of commands before allowing them
- Experiment with changes (requires specified goals)
  - Paired commands
    - ACTIVATE/DEACTIVATE
    - DEFINE/ATTACH/DETACH/DELETE
    - XAUTOLOG/FORCE
- Simplified version of reality

For paired commands, it should be possible to back out of an experiment that isn't panning out. For system changes with no obvious backout path, we can always just restart the model, but then we would have to wait awhile for it to IPL and stabilize before it can be used again for such testing.

“Simplified version of reality” – Sure, we could devote a whole LPAR to the model instead of running 2<sup>nd</sup>-level. But we'd lose portability, easy automation, low impact, etc. Modeling is always a trade-off between effort/expense and accuracy. The z/VM® operating system supplies tools (e.g. storage size, SET SHARE) to ensure our model does not exceed intended expense.

## Current Model

- Shows same (or nearly so) output as 1<sup>st</sup>-level for many QUERYs
  - Q NAMES
  - Q CPLOAD
  - Q CPLEVEL
  - Q USERID
  - Q DASD
- Can CPXLOAD or SHUTDOWN REIPL MODULE XXX
- Many other common environment-changing commands available, e.g. DEF VSWITCH

## Obvious Desirable Improvements

- Representative load for entire system
  - So that INDICATE output matches 1<sup>st</sup>-level
- Representative load for individual users
  - INDICATE USER matches (note that some proportionality may be involved)
- I/O and connectivity simulation
- Speculation engine for HOMUNCB
  - Generates proposed configuration changes
  - Implements proposed change on model
  - Evaluates results
  - Backs out change, makes 1<sup>st</sup>-level recommendation

## Try it yourself

- Code and further explanation is available via the zVM® Download Library
- <http://www.vm.ibm.com/download/packages/descript.cgi?HOMUNC>
- (Advertisement) There is lots of other good stuff on there too:
- <http://www.vm.ibm.com/download/packages/>

I currently have 12 packages available for download from the zVM® download library. Some are outdated, but we use CP1STLVL and CHUG every day. DR\_DRCT and RENSSI get only occasional use, but on those occasions have been very helpful. D26C gets used but usually needs updating for each round. And yes, HOMUNC is running on some of our systems.

## Summary

- The z/VM® Operating System is a great platform for self-modeling
- Building the self-model can be automated
- Using the self-model, the system can test in advance the systemic effects of a command
- The system can decline to invoke commands it recognizes as detrimental
- The self-model could be used in searching for configuration improvements

## For More Information ...

### Web sites:

- <http://www.vm.ibm.com/> -- zVM<sup>®</sup> product home Web page
- <http://www.vm.ibm.com/library> -- the online zVM<sup>®</sup> product Library
- <http://www.vm.ibm.com/education> -- presentations, classes and information

### Via mailing lists:

- [IBMTCP-L@VM.MARIST.EDU](mailto:IBMTCP-L@VM.MARIST.EDU)
- [IBMVM@LISTSERV.UARK.EDU](mailto:IBMVM@LISTSERV.UARK.EDU)
- [LINUX-390@VM.MARIST.EDU](mailto:LINUX-390@VM.MARIST.EDU)

### Contact Information:

Tim Greer  
z/VM<sup>®</sup> System Test  
[timgreer at us dot ibm dot com](mailto:timgreer@us.ibm.com)  
+1 607.429.3598