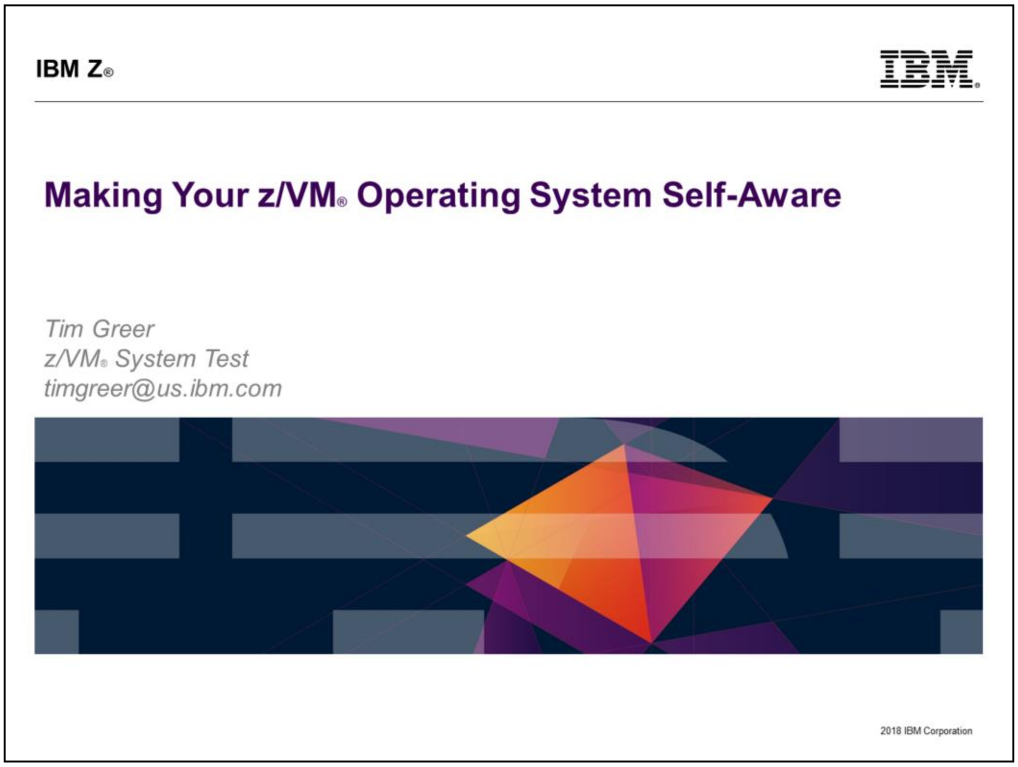


IBM Z®

Making Your z/VM® Operating System Self-Aware

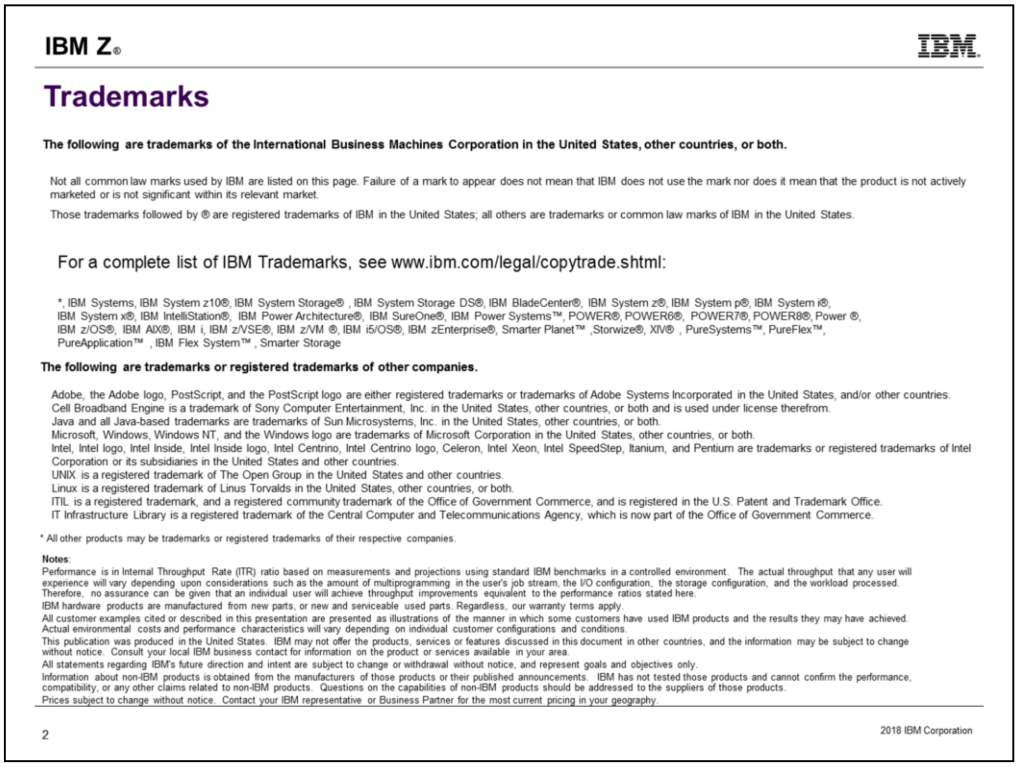
Tim Greer
z/VM® System Test
timgreer@us.ibm.com

• Last updated 2018 MAR 26 by Tim Greer.

Abstract:

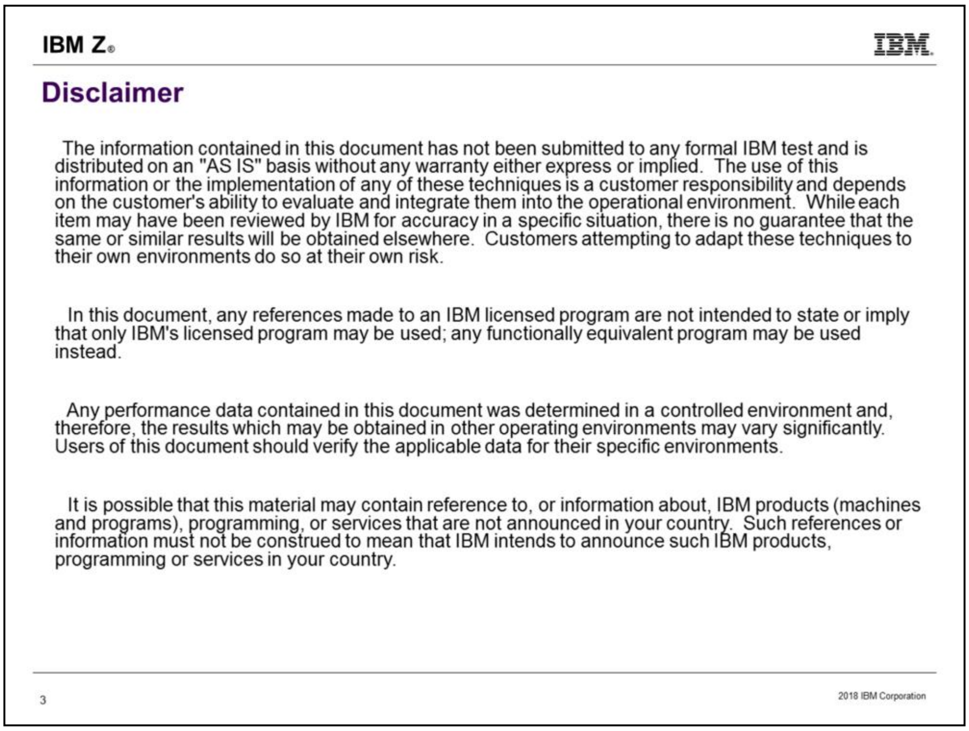
The z/VM® operating system's hypervising prowess makes it an ideal platform for self-awareness.

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.

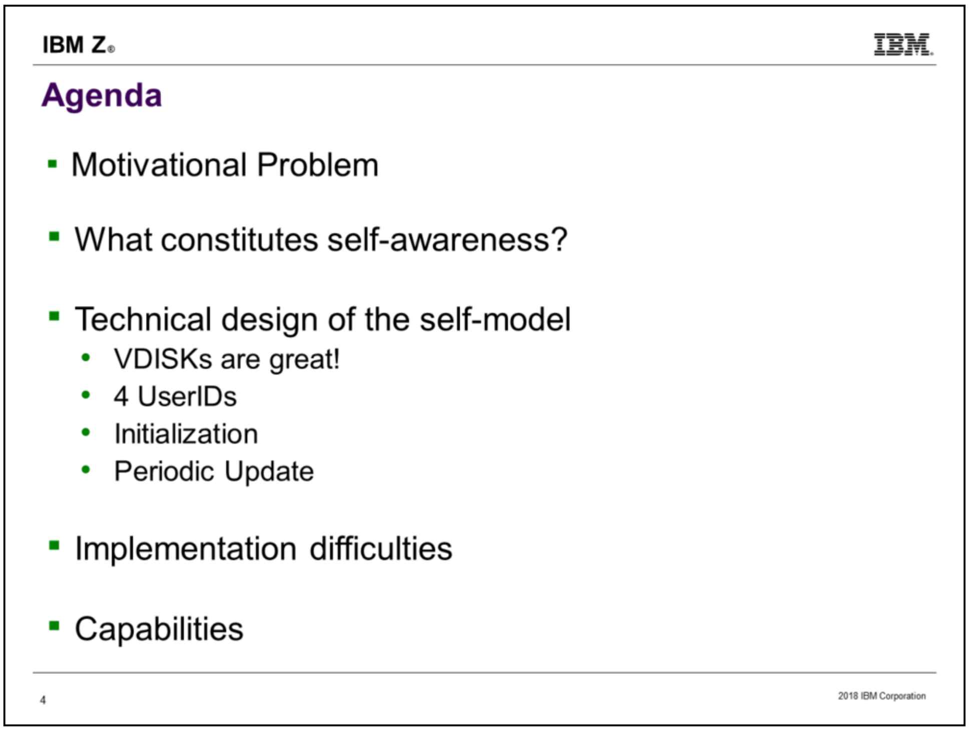


IBM Z®

Trademarks

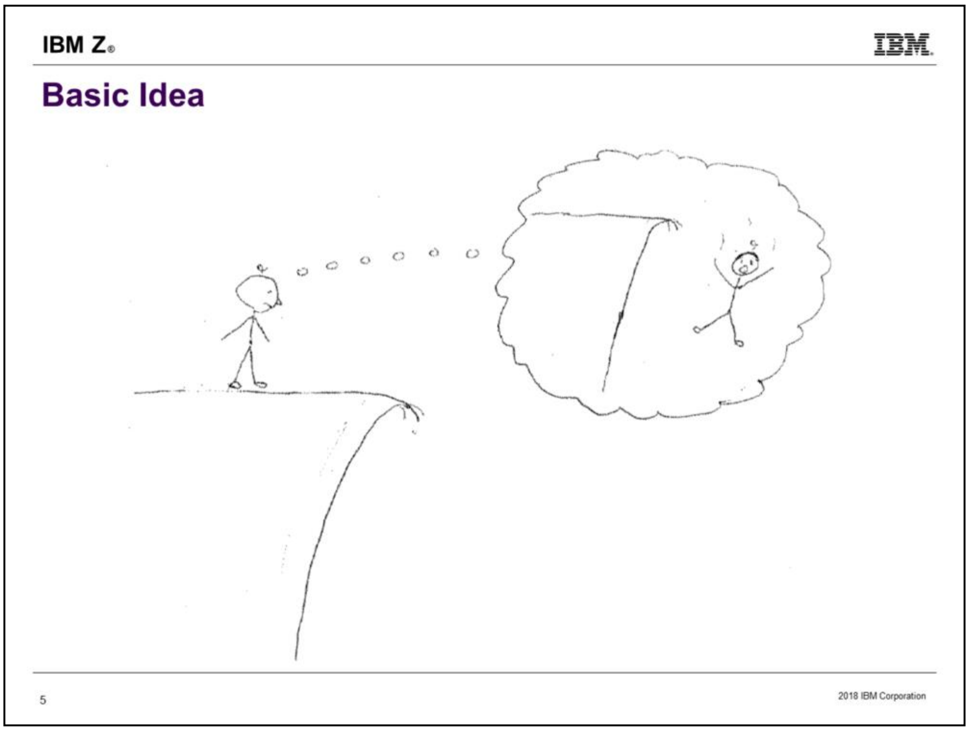


Disclaimer



Agenda

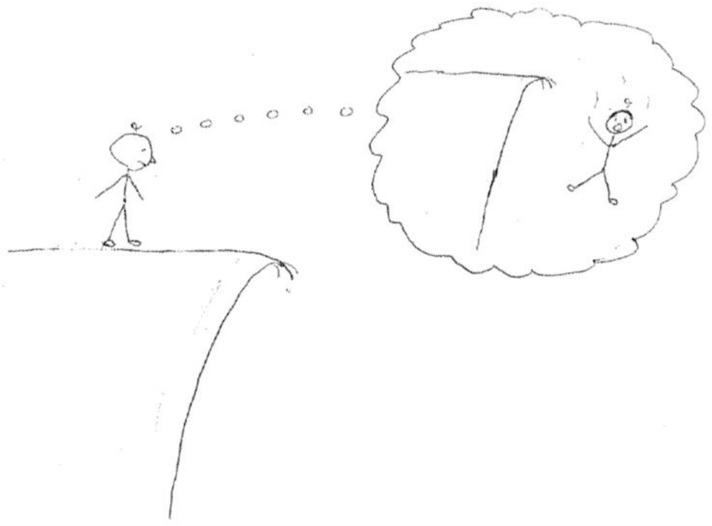
- Motivational Problem
- What constitutes self-awareness?
- Technical design of the self-model
  - VDISKs are great!
  - 4 UserIDs
  - Initialization
  - Periodic Update
- Implementation difficulties
- Capabilities



IBM Z®

IBM

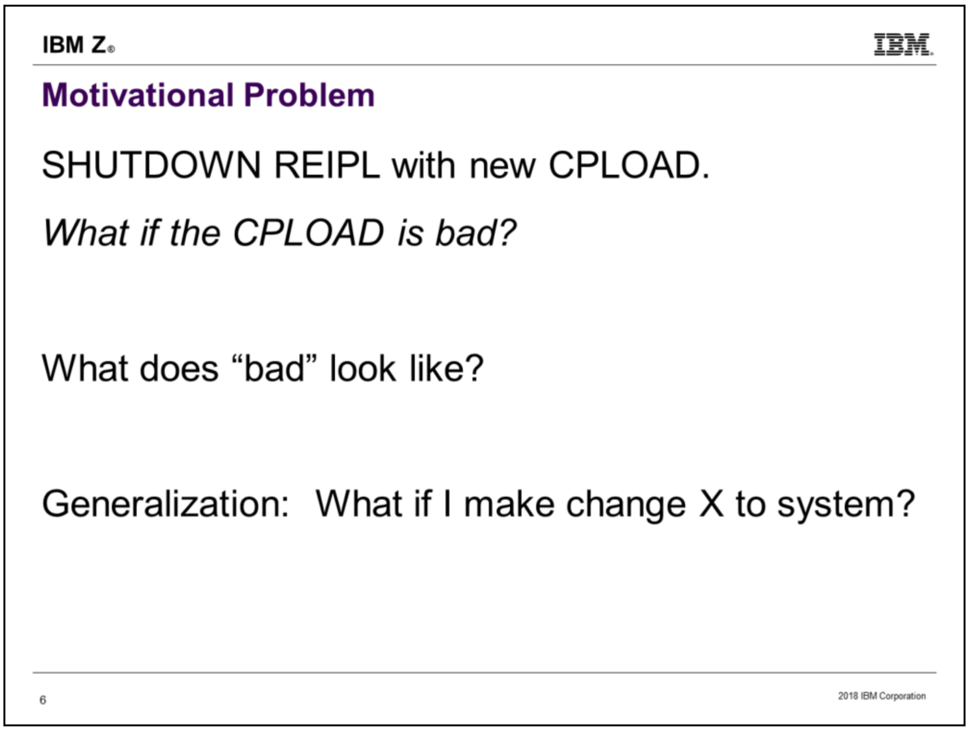
Basic Idea



5

2018 IBM Corporation

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.



IBM Z®

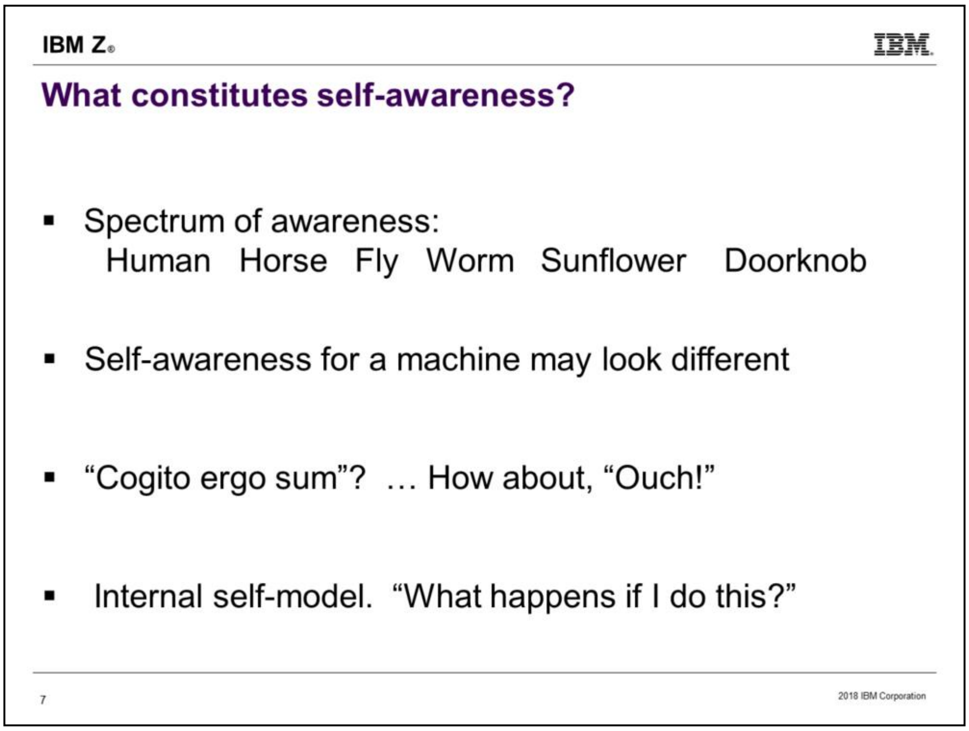
Motivational Problem

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?


IBM Z®

What constitutes self-awareness?

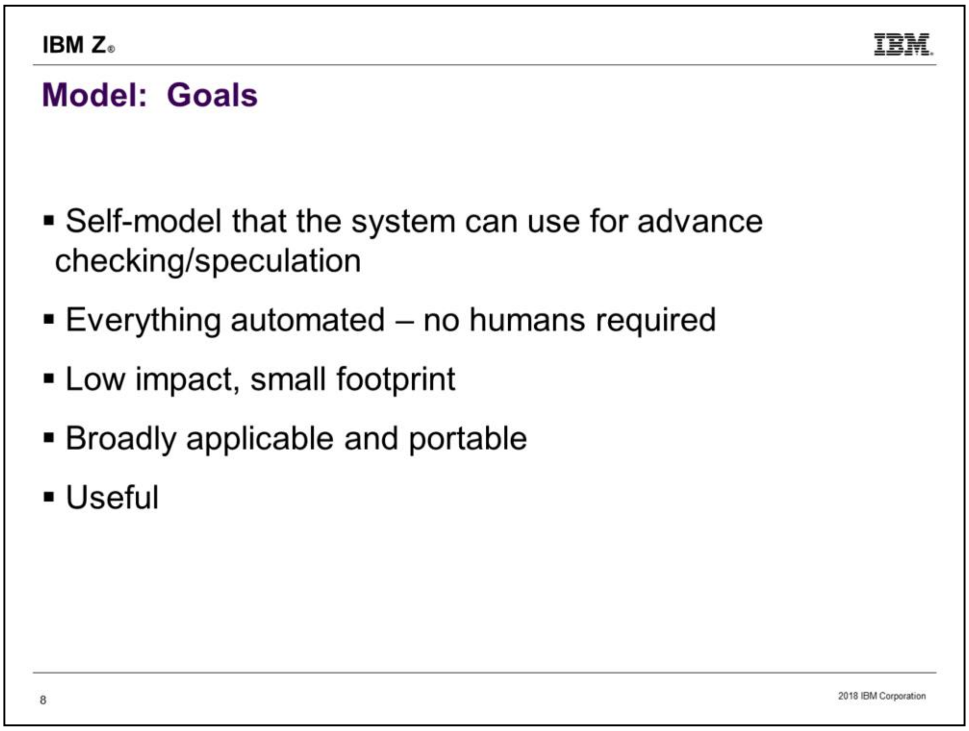
- Spectrum of awareness:
  Human Horse Fly Worm Sunflower Doorknob
- Self-awareness for a machine may look different
- “Cogito ergo sum”? … How about, “Ouch!”
- Internal self-model. “What happens if I do this?”

7 2018 IBM Corporation

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?”

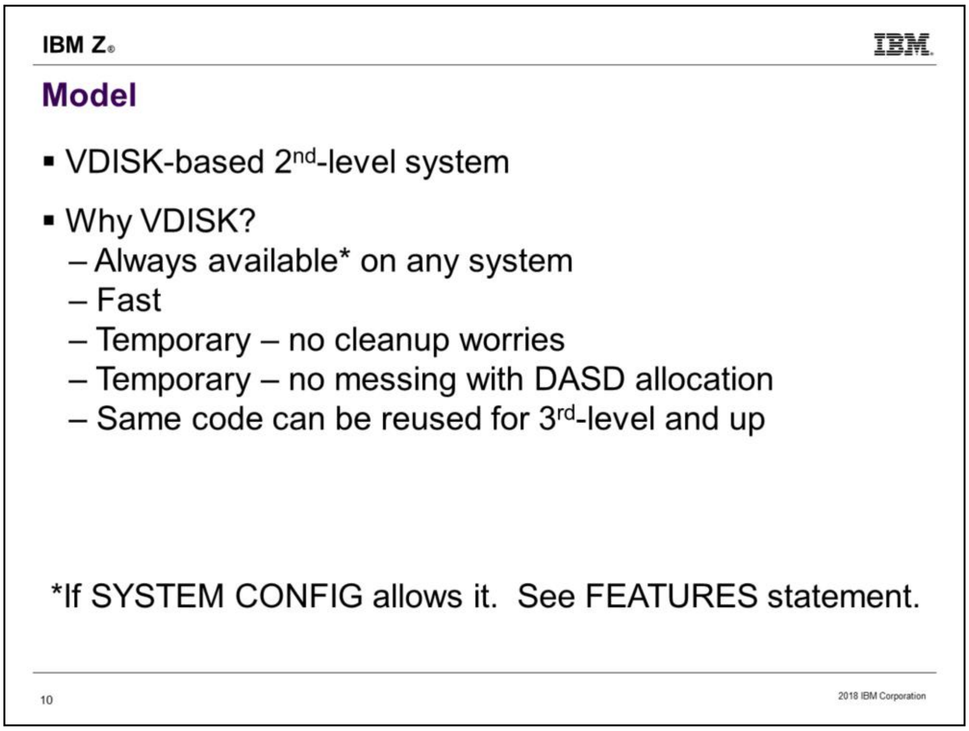
What about a self-model of an SSI? Analogous to self-awareness of a beehive.

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



IBM Z®

Model

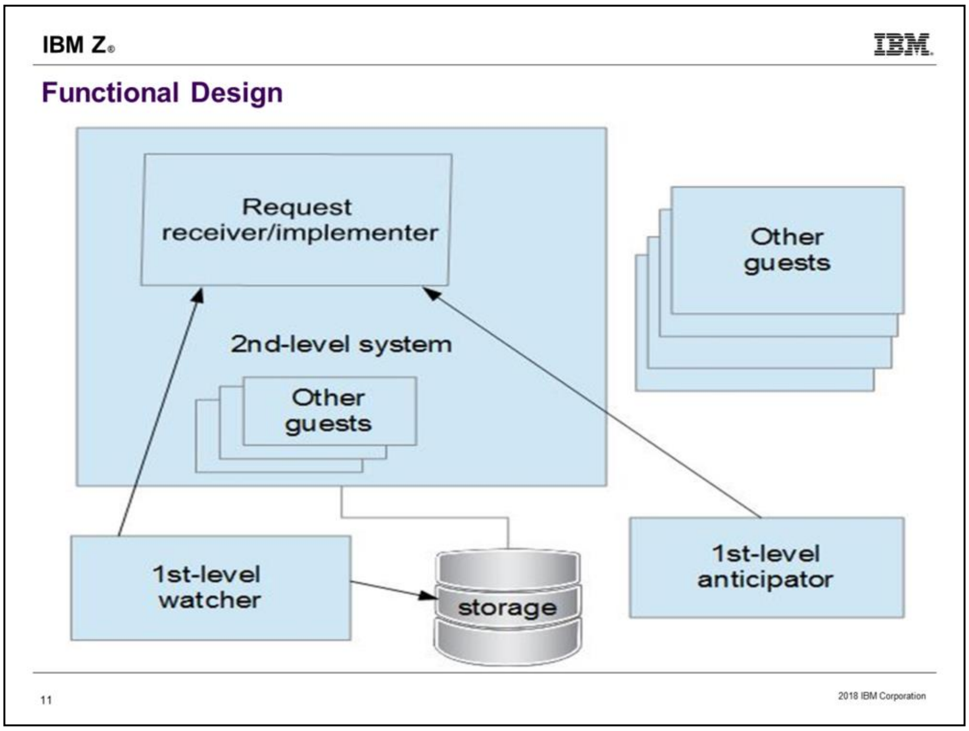
- VDISK-based 2nd-level system
- Why VDISK?
  - Always available* on any system
  - Fast
  - Temporary – no cleanup worries
  - Temporary – no messing with DASD allocation
  - Same code can be reused for 3rd-level and up

*If SYSTEM CONFIG allows it. See FEATURES statement.

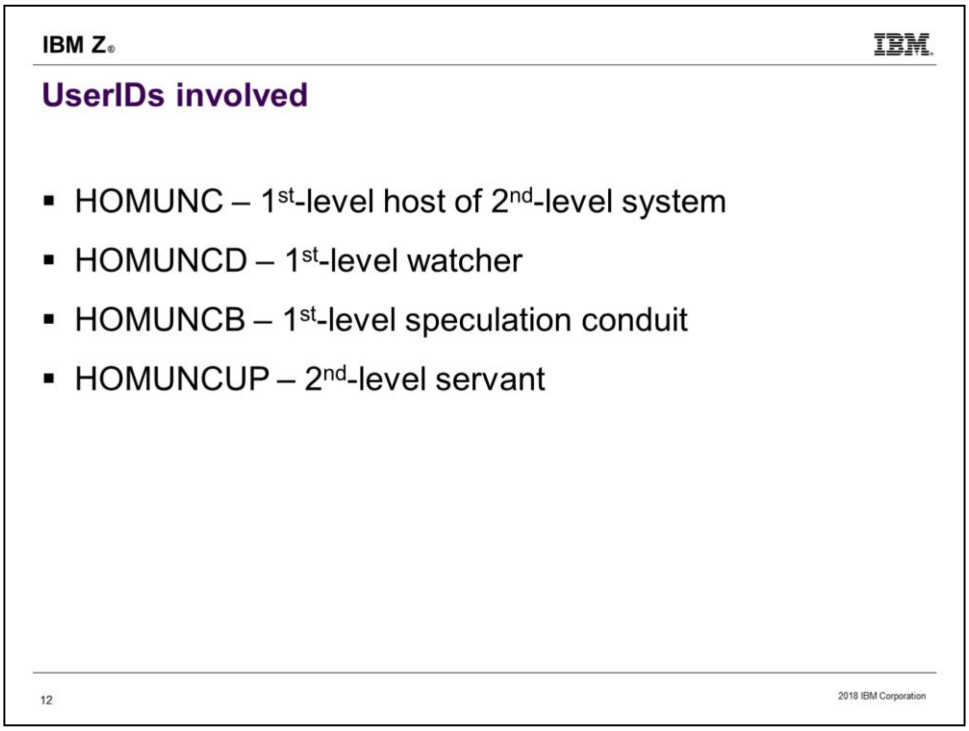
10

2018 IBM Corporation

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



Functional Design


IBM Z®

UserIDs involved

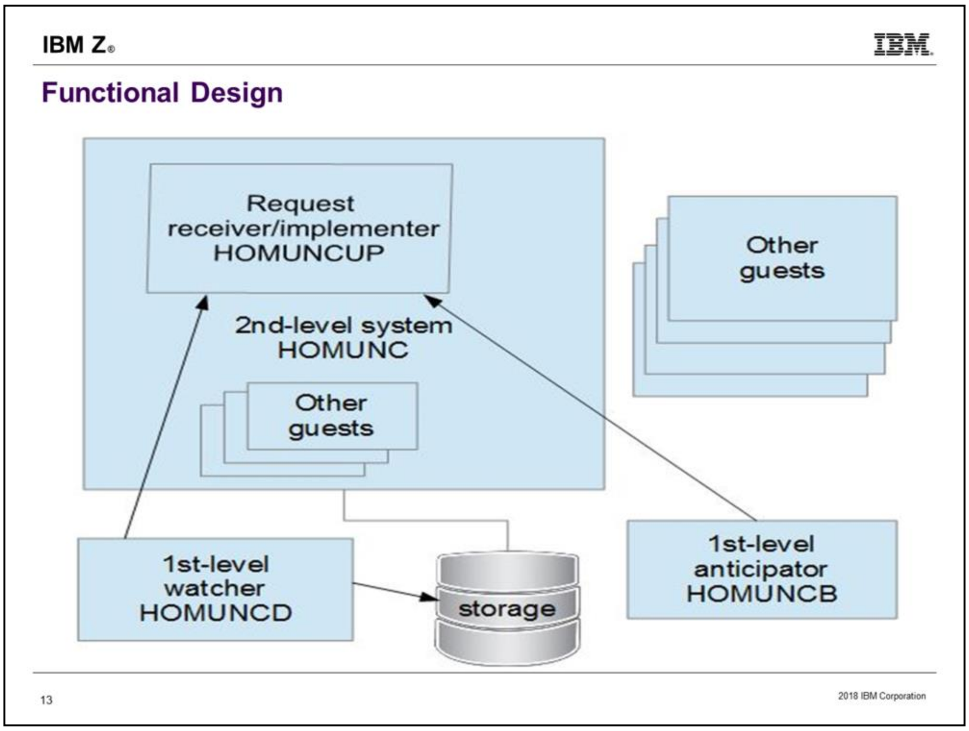
- HOMUNC – 1st-level host of 2nd-level system
- HOMUNCD – 1st-level watcher
- HOMUNCB – 1st-level speculation conduit
- HOMUNCUP – 2nd-level servant

12

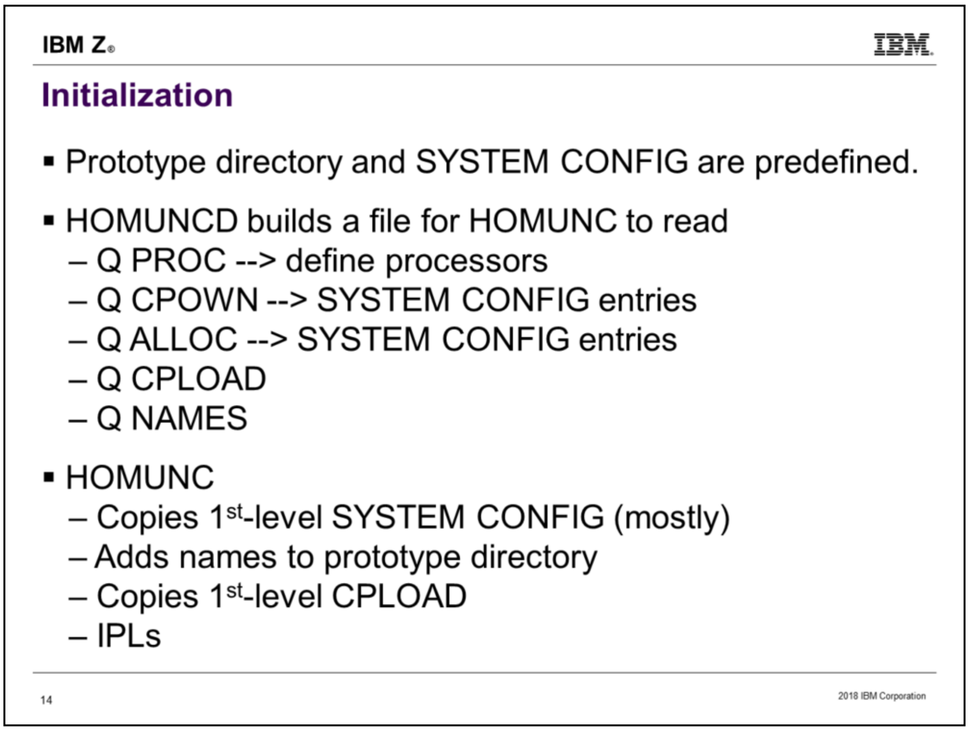
2018 IBM Corporation

HOMUNCD is the only 1st-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 1st-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.

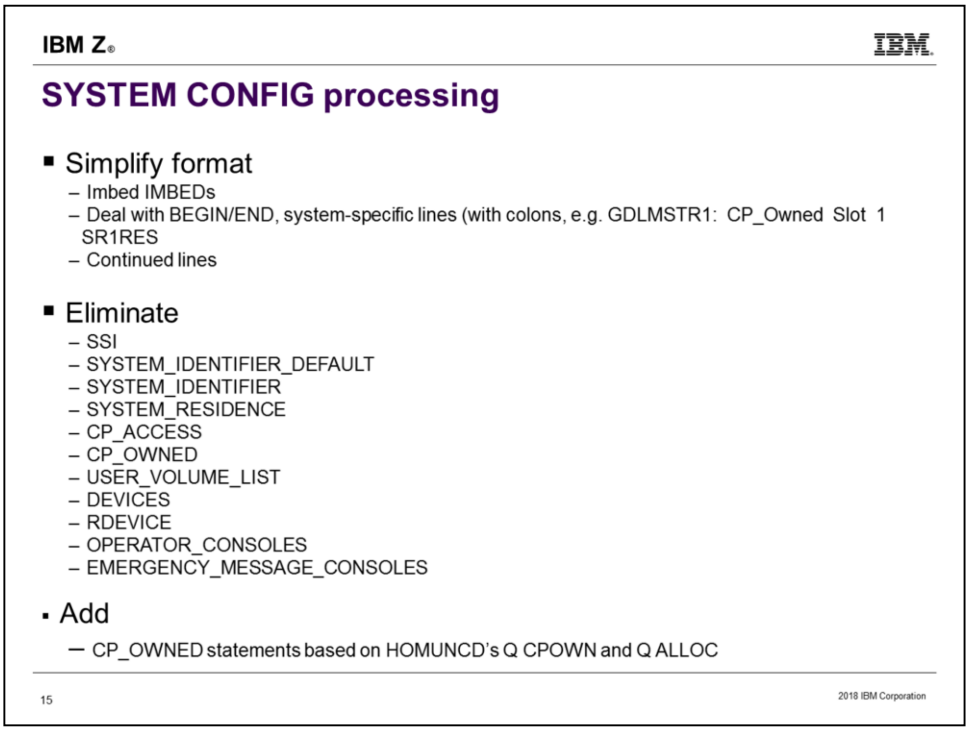


Same as previous slide, but with UserIDs added.



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 1st-level SYSTEM CONFIG (mostly)
  - Adds names to prototype directory
  - Copies 1st-level CPLOAD
  - IPLs



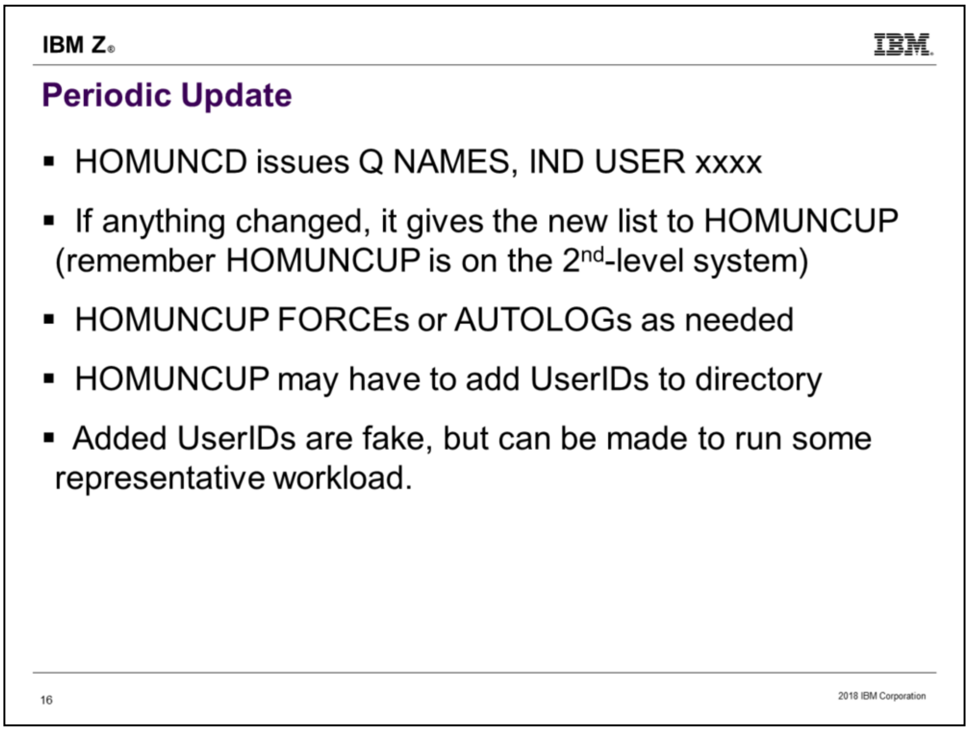
IBM Z®

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

15 2018 IBM Corporation

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



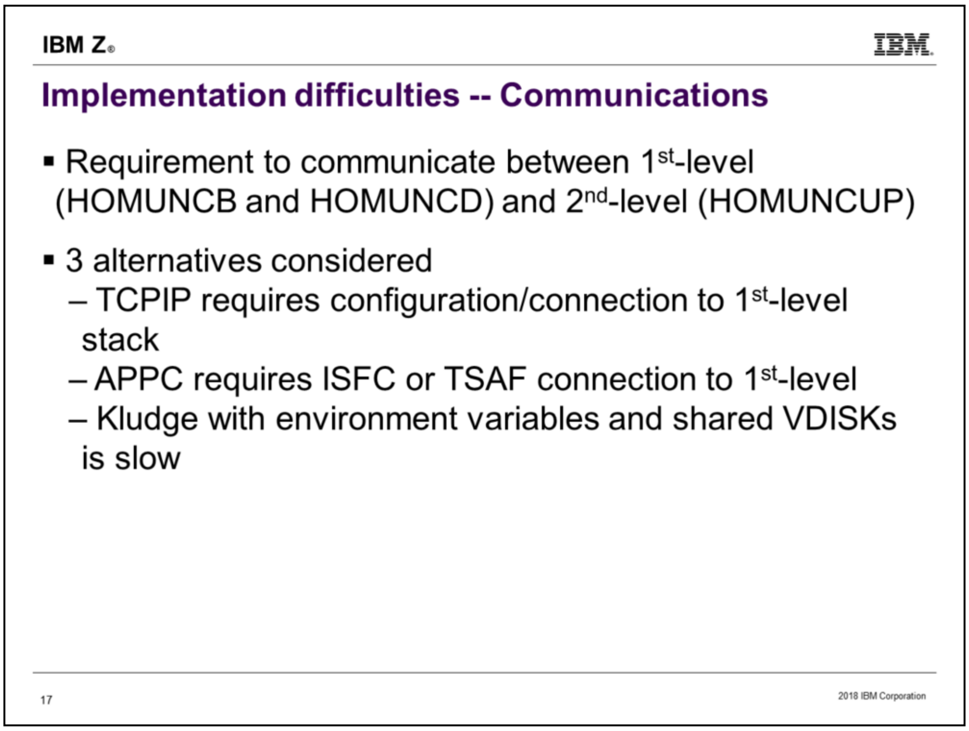
IBM Z®

Periodic Update

- HOMUNCD issues Q NAMES, IND USER xxxx
- If anything changed, it gives the new list to HOMUNCUP (remember HOMUNCUP is on the 2nd-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.

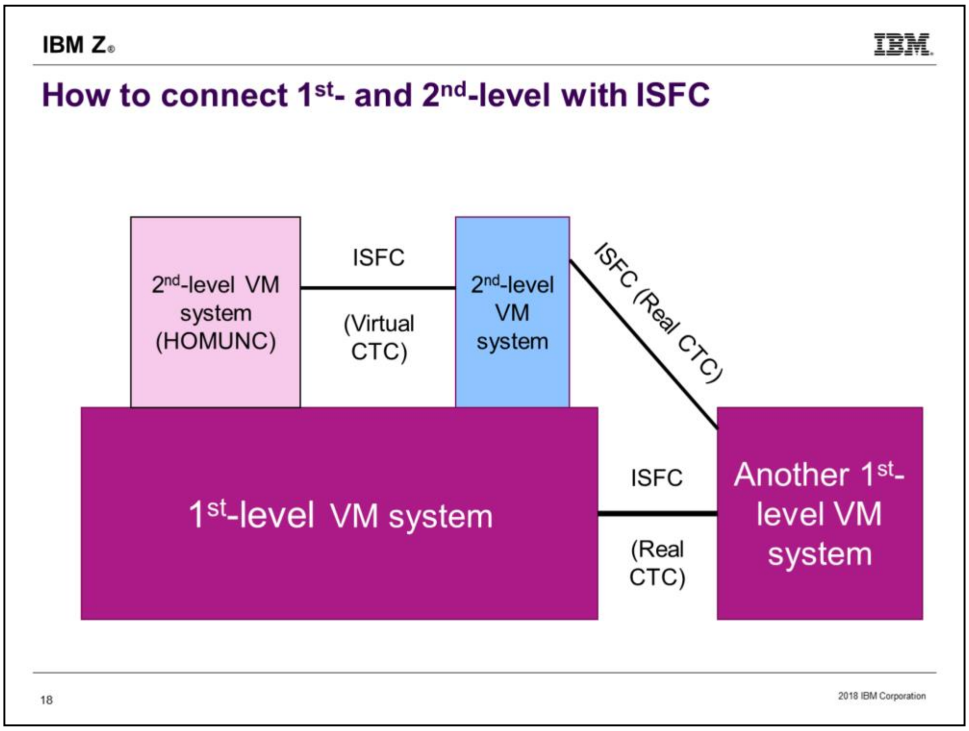
16 2018 IBM Corporation

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



Implementation difficulties -- Communications

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



IBM Z®

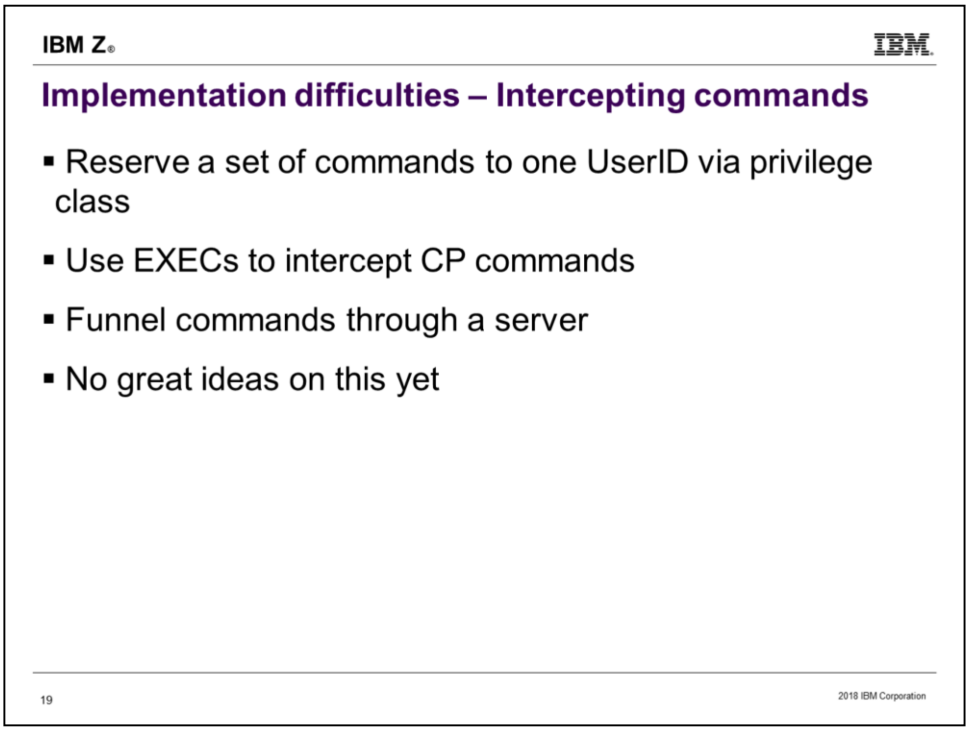
How to connect 1st- and 2nd-level with ISFC

18

2018 IBM Corporation

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

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


IBM Z®

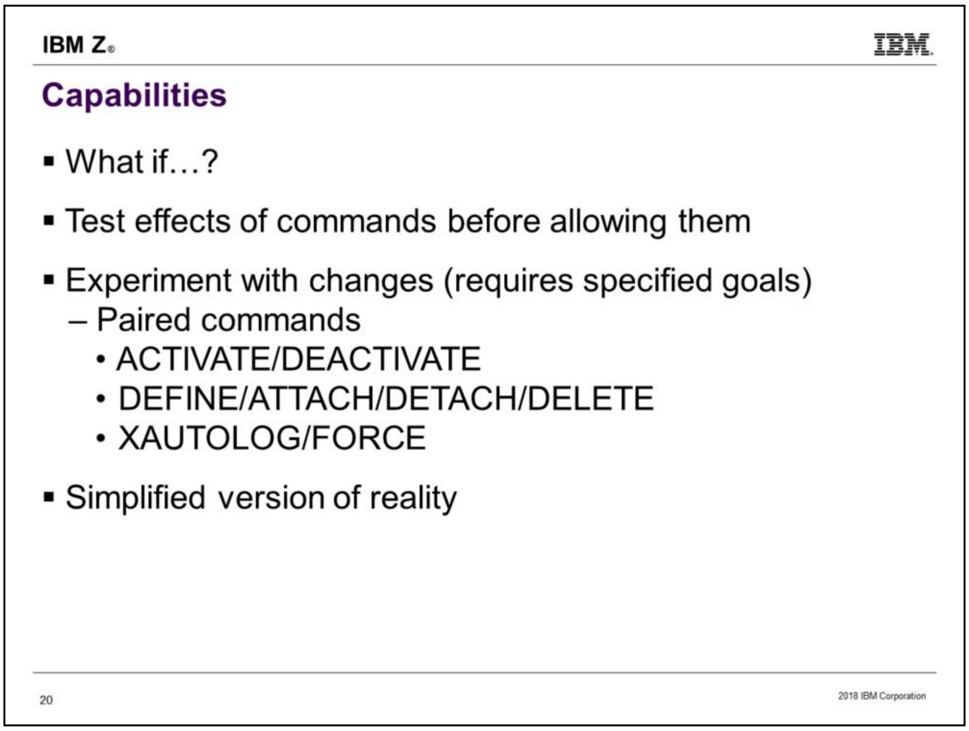
Implementation difficulties – Intercepting commands

- 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

19

2018 IBM Corporation

These are proposed methods of intercepting 1st-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.


IBM Z®

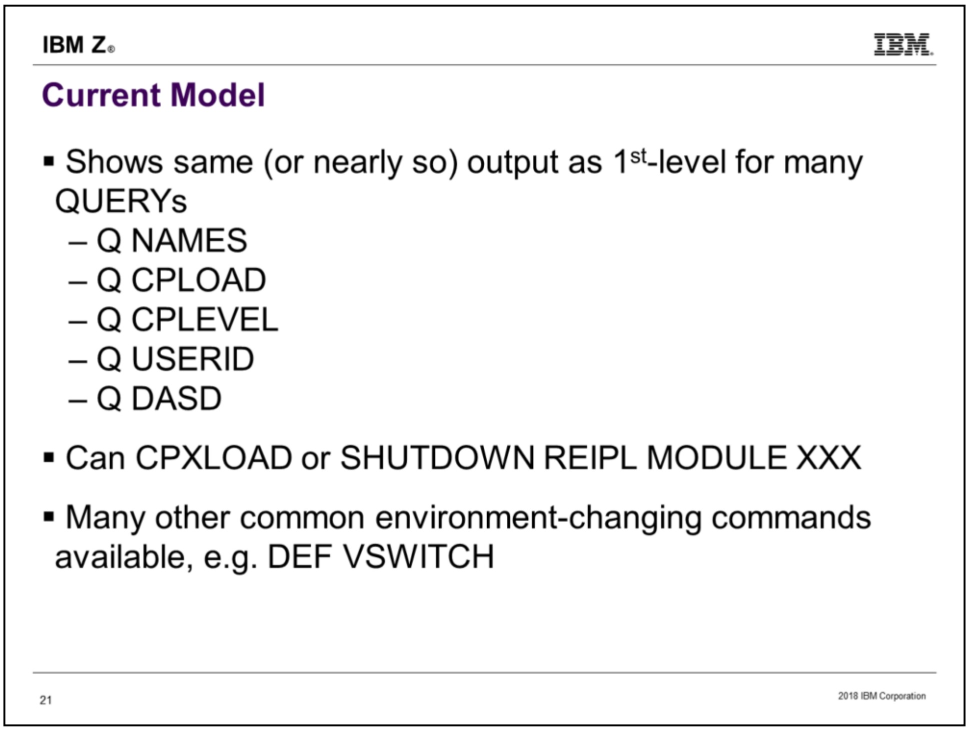
Capabilities

- 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

20 2018 IBM Corporation

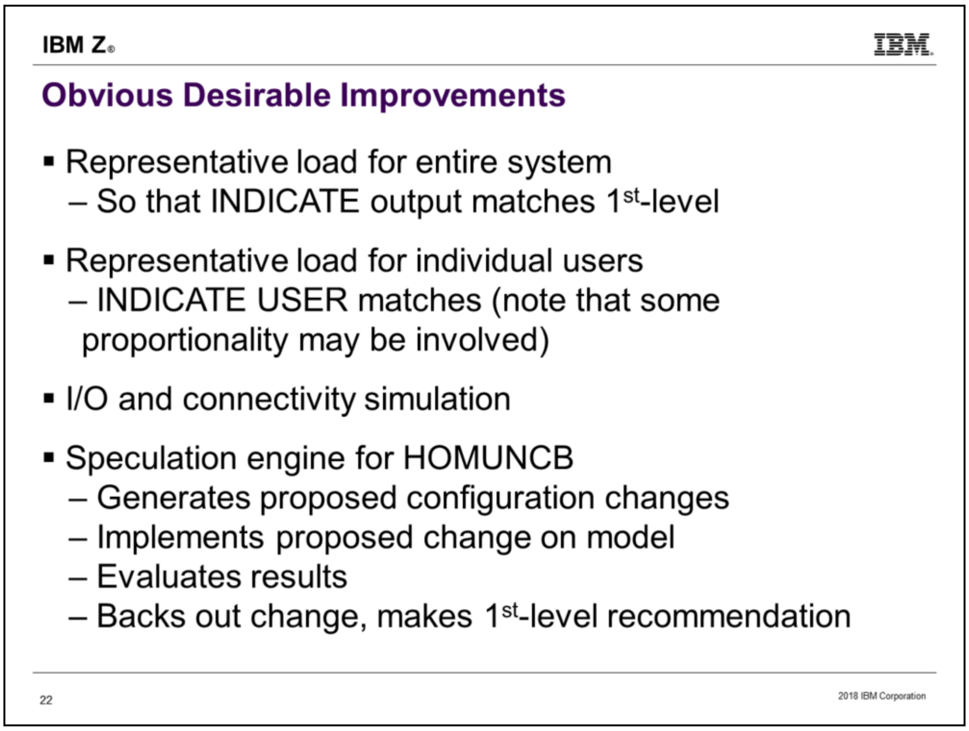
For paired commands, it should be possible to back out of an experiment that isn’t panning out. For system changes with no obvious backout path, we can always just restart the model, but then we would have to wait awhile for it to IPL and stabilize before it can be used again for such testing.

“Simplified version of reality” – Sure, we could devote a whole LPAR to the model instead of running 2nd-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.



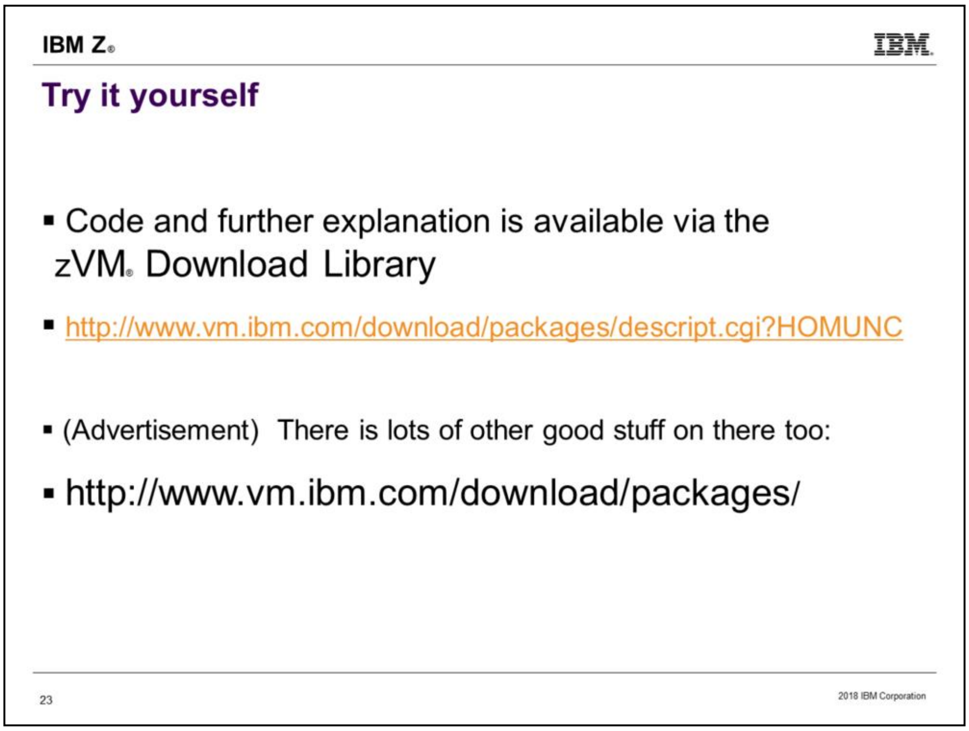
Current Model

- Shows same (or nearly so) output as 1st-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 1st-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 1st-level recommendation



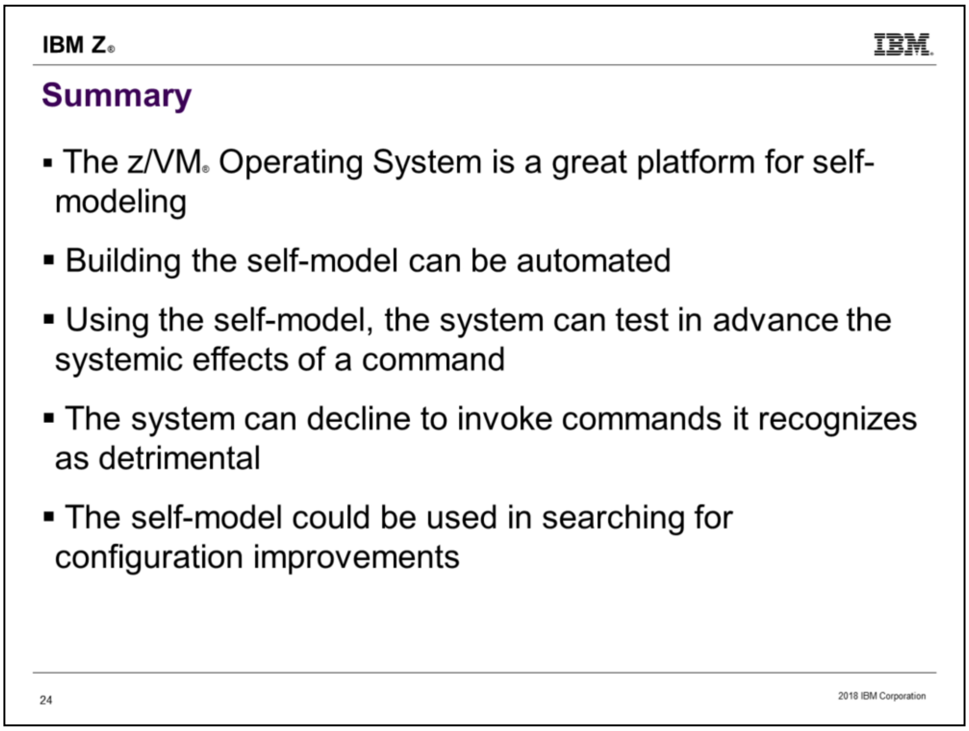
IBM Z®

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/

23 2018 IBM Corporation

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

Via mailing lists:
- IBMTCP-L@VM.MARIST.EDU
- IBMVM@LISTSERV.UARK.EDU
- LINUX-390@VM.MARIST.EDU

Contact Information:

Tim Greer
z/VM® System Test
timgreer at us dot ibm dot com
+1 607.429.3598