HP-UX Kernel Tuning Rearchitecture

Steven Roth
Hewlett-Packard
August 24, 2001

Hewlett-Packard is making major improvements to the way the HP-UX kernel is tuned, including:

dynamic tunables
self-tuning algorithms
no kernel rebuilds for tuning
improved tuning defaults
and more

HP-UX Kernel Tuning Rearchitecture

Steven Roth
Hewlett-Packard
August 24, 2001

overview of HP-UX kernel tuning

- What are tunables?
- What do they control?
- How are they changed?

overview of HP-UX kernel tuning

What Are Tunables?

The HP-UX kernel has a set of configurable parameters called “tunables.” There are about 150 of them in HP-UX 11i, such as:

nproc
maxdsiz
dbc_max_pct
shmmax

overview of HP-UX kernel tuning

What Do They Control?

Tunables are used for a variety of purposes, including:

- Sizing kernel data structures
- Limiting process resource usage
- Enabling optional features
- Tailoring kernel performance

Some tunables are intended for use only within HP.

overview of HP-UX kernel tuning

How Are They Changed?

Tunables are changed using the "Kernel Configurable Parameters" screen in SAM,

or by using the *kmtune*(1M) command,

or by calling the *settune*(2) system call.

Although not a supported method, some customers also used to change tunables by editing the /stand/system file and manually rebuilding their kernel.

what is changing?

- dynamic changes of tunables
- self-tuning tunables
- no kernel rebuilds for tuning
- algorithmic default values
- man pages for tunables
- more error checks and warnings
- self-descriptive names
- obsolescence of tunables
- two-tier visibility of tunables
- tunable sets
- meta-tunables
- driver instance tunables
- access through DMI and WBEM

what is changing?

- dynamic changes of tunables
- self-tuning tunables
- no kernel rebuilds for tuning
- algorithmic default values
- man pages for tunables
- more error checks and warnings
- self-descriptive names
- obsolescence of tunables
- two-tier visibility of tunables
- tunable sets
- meta-tunables
- driver instance tunables
- access through DMI and WBEM

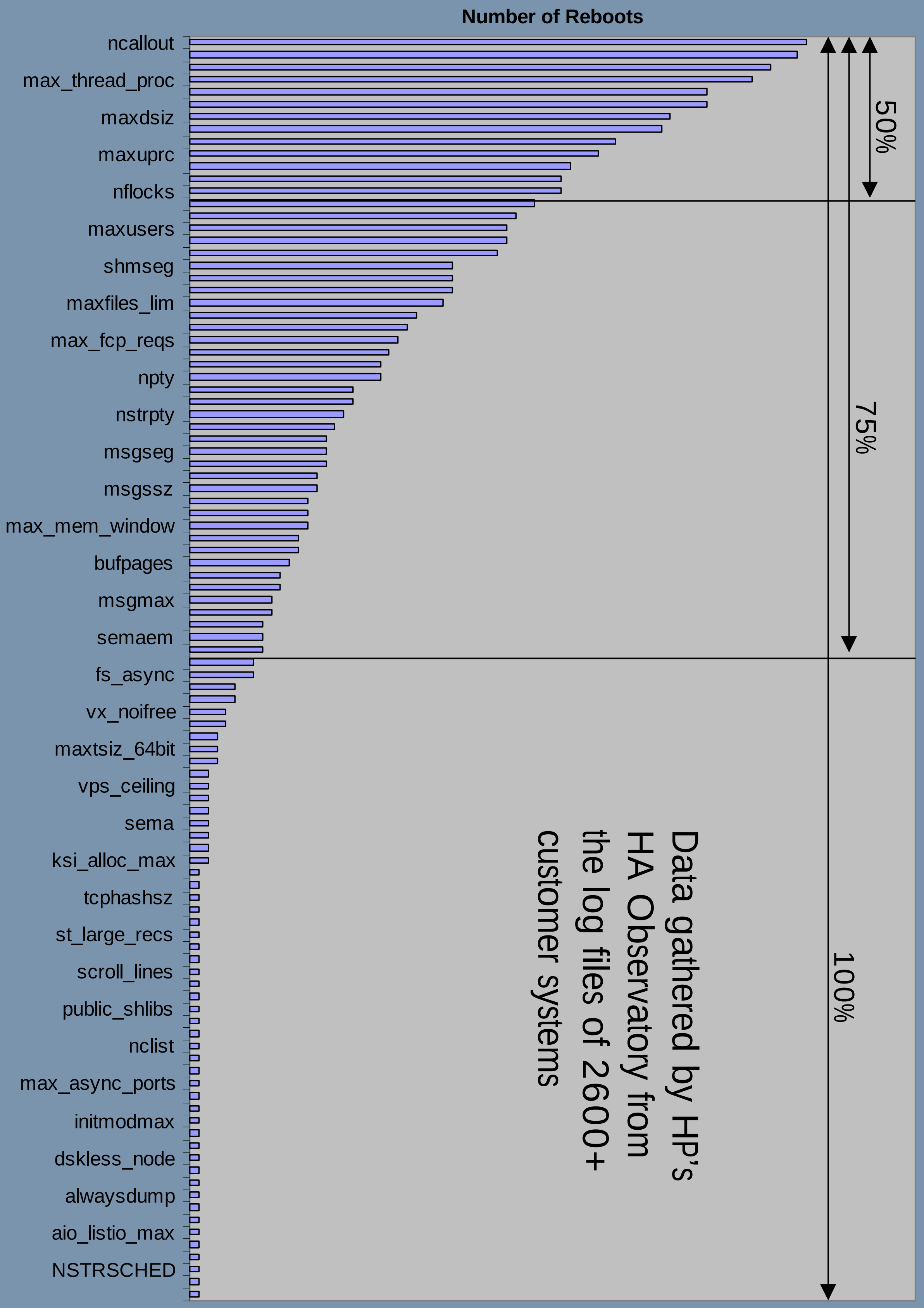
and why?

Dynamic Changes of Tunables

- 12 tunables made dynamic in 11i (can change without reboot)
- More tunables made dynamic in each subsequent release, starting with those changed most often
- Change data comes from analyzing customer log files

Having to reboot to change tunables is frustrating, reduces system availability, and inhibits performance tuning.

Number of Reboots for Tuning



what is changing?

- dynamic changes of tunables
- self-tuning tunables
- no kernel rebuilds for tuning
- algorithmic default values
- man pages for tunables
- more error checks and warnings
- self-descriptive names
- obsolescence of tunables
- two-tier visibility of tunables
- tunable sets
- meta-tunables
- driver instance tunables
- access through DMI and WBEM

and why?

Self-Tuning Tunables

- Tunables which continually re-tune themselves
- System adapts automatically to changing conditions
- Admin can override and force a specific value

Some tunables, such as those controlling sizes of some kernel data structures, should not need to be tuned by an administrator except under unusual conditions.

what is changing?

- dynamic changes of tunables
- self-tuning tunables
- no kernel rebuilds for tuning
- algorithmic default values
- man pages for tunables
- more error checks and warnings
- self-descriptive names
- obsolescence of tunables
- two-tier visibility of tunables
- tunable sets
- meta-tunables
- driver instance tunables
- access through DMI and WBEM

and why?

No Kernel Rebuilds for Tuning

- Tunable values kept in Kernel Registry database
- Kernel reads tunable values from Registry at boot time, with failsafe backup
- Dynamic changes to tunable values are written back to Kernel Registry in real time
- No tuning information in /stand/system, master files, space files, or compiled kernels

Rebuilding a kernel just to change a tunable is inefficient. Having tunable values compiled into the kernel makes no sense for dynamic or self-tuning tunables.

About the Kernel Registry

- Used by the kernel to keep persistent data across reboots
- Tunable data is only one of many things stored in it
- Similar in concept to the Windows Registry
- No direct user interface:
  Access tunable data through SAM, *kmtune*(1M), or *tuneinfo*(2)
- Automatically restores "last known good" copy in case of corruption or other failure
- If even that doesn't work, kernel will boot using "fail-safe" values for all tunables, so that system can be repaired

what is changing?

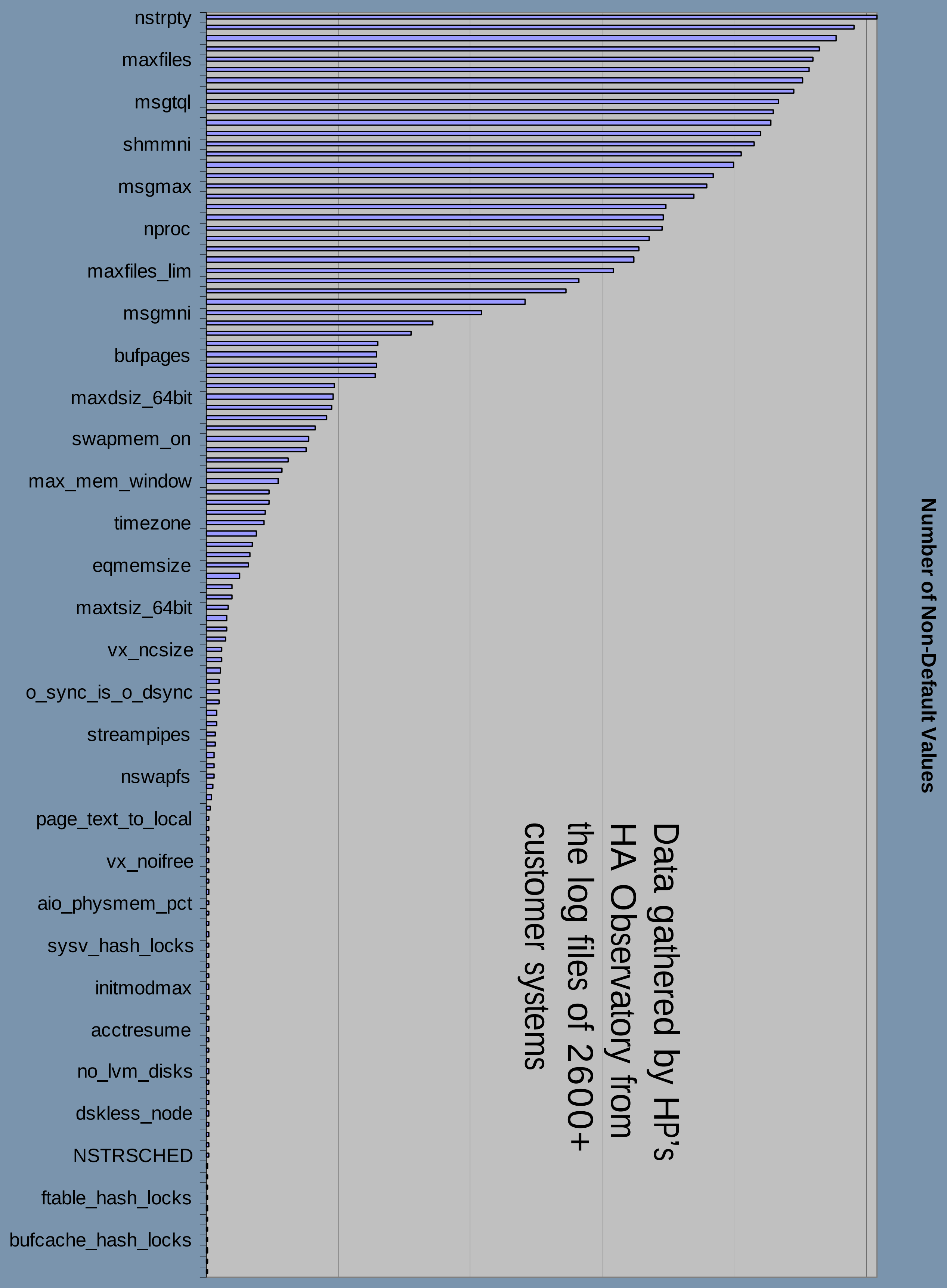
- dynamic changes of tunables
- self-tuning tunables
- no kernel rebuilds for tuning
- algorithmic default values
- man pages for tunables
- more error checks and warnings
- self-descriptive names
- obsolescence of tunables
- two-tier visibility of tunables
- tunable sets
- meta-tunables
- driver instance tunables
- access through DMI and WBEM

and why?

Algorithmic Default Values

- Default values of tunables updated
- Updated values come from analyzing customer log files
- Some defaults chosen at boot time based on system size, type, etc.
- Some limit tunables may default to “unlimited”

Customer data tells us our old default values were too restrictive. Using boot-time algorithms we no longer need a one-size-fits-all default value.


Number of Non-Default Values
Data gathered by HP's HA Observatory from the log files of 2600+ customer systems
nstrpty
maxfiles
msgtql
shmmni
msgmax
nproc
maxfiles_lim
msgmni
bufpages
maxdsiz_64bit
swapmem_on
max_mem_window
timezone
eqmemsize
maxtsiz_64bit
vx_ncsize
o_sync_is_o_dsync
streampipes
nswapfs
page_text_to_local
vx_noifree
aio_physmem_pct
sysv_hash_locks
initmodmax
acctresume
no_lvm_disks
dskless_node
NSTRSCHED
ftable_hash_locks
bufcache_hash_locks

what is changing?

- dynamic changes of tunables
- self-tuning tunables
- no kernel rebuilds for tuning
- algorithmic default values
- man pages for tunables
- more error checks and warnings
- self-descriptive names
- obsolescence of tunables
- two-tier visibility of tunables
- tunable sets
- meta-tunables
- driver instance tunables
- access through DMI and WBEM

and why?

Man Pages for Tunables

- Every tunable has a man page
- Man Pages give guidance on when to change values, what values to choose, and what side-effects will occur
- Man pages will be available through *man*(1), SAM, and on docs.hp.com

Some customers have reported difficulty finding or using our tuning documentation.

Tunable Man Pages

Each man page has the following headings:

- Values
- Description
- Who is expected to change this tunable?
- Restrictions on Changing
- When should the value of this tunable be raised?
- What are the side-effects of raising the value?
- When should the value of this tunable be lowered?
- What are the side-effects of lowering the value?
- What other tunables should be changed at the same time?

what is changing?

- dynamic changes of tunables
- self-tuning tunables
- no kernel rebuilds for tuning
- algorithmic default values
- man pages for tunables
- more error checks and warnings
- self-descriptive names
- obsolescence of tunables
- two-tier visibility of tunables
- tunable sets
- meta-tunables
- driver instance tunables
- access through DMI and WBEM

and why?

More Error Checks and Warnings

- Tunable changes will be checked to ensure validity and self-consistency
- Tunable changes will be validated against current system configuration and usage
- Errors result in informative English error messages
- Valid but suspicious changes will result in informative warning messages

Having the kernel silently ignore bad values is confusing. Numeric error codes and jargon messages are unhelpful. Warning messages can prevent unwanted mistakes.

what is changing?

- dynamic changes of tunables
- self-tuning tunables
- no kernel rebuilds for tuning
- algorithmic default values
- man pages for tunables
- more error checks and warnings
- self-descriptive names
- obsolescence of tunables
- two-tier visibility of tunables
- tunable sets
- meta-tunables
- driver instance tunables
- access through DMI and WBEM

and why?

Self-Descriptive Names

- Longer, self-descriptive names chosen for all tunables
- New names use fewer abbreviations and include units.
- Old names still work as aliases for compatibility

Older tunable names are cryptic: difficult to understand, use, and remember.

Self-Descriptive Names

Old Name	New Name
nproc	max_processes_per_system
maxuprc	max_processes_per_user
semmns	max_sysV_semaphores
hfs_max_ra_blocks	max_readahead_blocks_on_hfs
maxdsiz	max_32bit_process_data_bytes
default_disk_ir	enable_scsi_immediate_reporting
maxvgs	max_lvm_volume_groups
ninode	num_cached_inodes

what is changing?

- dynamic changes of tunables
- self-tuning tunables
- no kernel rebuilds for tuning
- algorithmic default values
- man pages for tunables
- more error checks and warnings
- self-descriptive names
- obsolescence of tunables
- two-tier visibility of tunables
- tunable sets
- meta-tunables
- driver instance tunables
- access through DMI and WBEM

and why?

Obsolescence of Tunables

- Tunables no longer used by the kernel are removed
- Tunables never used by any customers are removed
- Tunables controlling resources that no longer need admin control are removed
- Tunables used only within HP are no longer visible

Removing obsolete tunables reduces clutter in the list of available tunables, which makes it easier to find the tunables you need.

what is changing?

- dynamic changes of tunables
- self-tuning tunables
- no kernel rebuilds for tuning
- algorithmic default values
- man pages for tunables
- more error checks and warnings
- self-descriptive names
- obsolescence of tunables
- two-tier visibility of tunables
- tunable sets
- meta-tunables
- driver instance tunables
- access through DMI and WBEM

and why?

Two-Tier Visibility of Tunables

- SAM initially displays only commonly changed tunables
- The rest of the tunables are on a separate screen in SAM
- Tunables that have been changed on your system always show up in the first list
- *kmtune*(1M) displays all tunables

Showing only common and locally changed tunables makes it easier to find the tunables you use most. The others are still available when you need them.

what is changing?

- dynamic changes of tunables
- self-tuning tunables
- no kernel rebuilds for tuning
- algorithmic default values
- man pages for tunables
- more error checks and warnings
- self-descriptive names
- obsolescence of tunables
- two-tier visibility of tunables
- tunable sets
- meta-tunables
- driver instance tunables
- access through DMI and WBEM

and why?

Tunable Sets

- SAM can save the values of some or all tunables as a set
- The entire set can be applied in one simple, atomic operation
- Multiple sets can be created, giving an easy way to switch between tuning configurations

In the past, customers have kept several different kernels compiled with different tunable values, in order to change tuning “easily.” Tunable sets will make changing configurations simple.

what is changing?

- dynamic changes of tunables
- self-tuning tunables
- no kernel rebuilds for tuning
- algorithmic default values
- man pages for tunables
- more error checks and warnings
- self-descriptive names
- obsolescence of tunables
- two-tier visibility of tunables
- tunable sets
- meta-tunables
- driver instance tunables
- access through DMI and WBEM

and why?

Meta-Tunables

- Meta-tunables control the values of groups of other tunables
- Changing one high-level meta-tunable can result in changes to many low-level tunables
- Admin does not have to know all of the low-level tuning details
- Example: the expected number of users of a system

Meta-tunables further streamline the tuning process, by automating the low-level changes needed to effect a significant configuration change.

what is changing?

- dynamic changes of tunables
- self-tuning tunables
- no kernel rebuilds for tuning
- algorithmic default values
- man pages for tunables
- more error checks and warnings
- self-descriptive names
- obsolescence of tunables
- two-tier visibility of tunables
- tunable sets
- meta-tunables
- driver instance tunables
- access through DMI and WBEM

and why?

Driver Instance Tunables

- Device driver tunables can have a different value for each device controlled by that driver
- Each device inherits tunable values from the main driver settings unless overridden

Allowing each device to have its own tuning increases flexibility and allows for better performance.

what is changing?

- dynamic changes of tunables
- self-tuning tunables
- no kernel rebuilds for tuning
- algorithmic default values
- man pages for tunables
- more error checks and warnings
- self-descriptive names
- obsolescence of tunables
- two-tier visibility of tunables
- tunable sets
- meta-tunables
- driver instance tunables
- access through DMI and WBEM

and why?

Access through DMI and WBEM

- Tunable information can be viewed and changed through industry standard management interfaces
- DMI is the Distributed Management Interface
- WBEM is the Web-Based Enterprise Management standard
- Various system and enterprise management tools can use these interfaces

System tuning can be controlled through industry standard, platform neutral interfaces. This makes multi-platform enterprise management easier.

how will the changes affect me?

- easier to tune your system
- adopt new default values
- no tunable info in system file
- third-party tunables

how will the changes affect me?

Easier to Tune Your System

A well-tuned system will be easier to achieve because:

- Much of it is done automatically
- Reboots will rarely be needed
- Kernel doesn’t have to be rebuilt
- Tunables are well documented and named, and error and warning messages are helpful

how will the changes affect me?

Adopt New Default Values

The new, algorithmic default values may provide better service than your present hard-coded values.

Any tuning you have already done will not be replaced automatically.

You may choose to remove your hard-coded values so that the algorithmic default values and self-tuning tunables can deliver optimal performance.

how will the changes affect me?

No Tuning Info in System File

There will no longer be any tuning information in /stand/system. Any administrators, scripts, or applications which use that file to obtain or change tunable values must change to use a supported interface, such as:

- SAM
- WBEM/DMI
- *kmtune*(1M)
- *gettune*(2), *settune*(2), *settune_txn*(2), *tuneinfo*(2)
- *pstat*(2)

how will the changes affect me?

Third-Party Tunables

If you are a developer of device drivers or kernel modules for HP-UX, your code can have its own tunables, with all of the same capabilities as core HP-UX tunables.

Developer information should be available in late 2001. Please contact us if you are interested, at the address

dyntune-feedback@cup.hp.com

when is it coming?

<u>HP-UX 11i — mid-2000</u>

- dynamic tunables infrastructure
- 12 dynamic tunables

<u>between releases — fall 2001</u>

- man pages for tunables available as a patch and on docs.hp.com

<u>next kernel release — mid-2002</u>

- dynamic and self-tuning tunables
- no kernel rebuilds for tuning
- algorithmic default values
- self-descriptive names
- driver instance tunables
- etc.

<u>subsequent releases</u>

- meta-tunables
- tunable sets
- more dynamic and self-tuning tunables

feedback and questions

If you have any feedback or questions about HP-UX kernel tuning, please send them to

dyntune-feedback@cup.hp.com

Please note that the volume of feedback may prevent us from replying to every message.