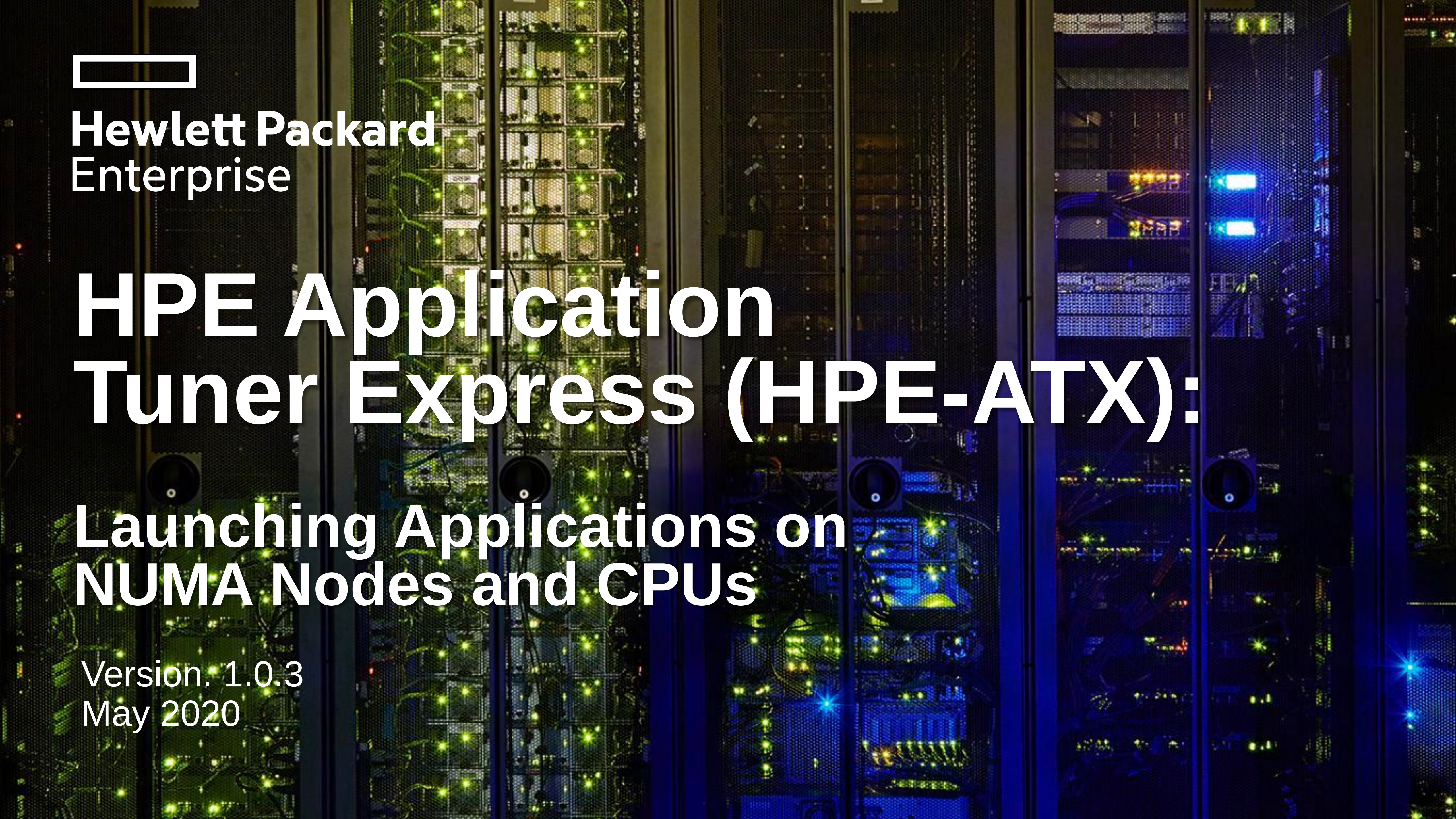


Hewlett Packard Enterprise

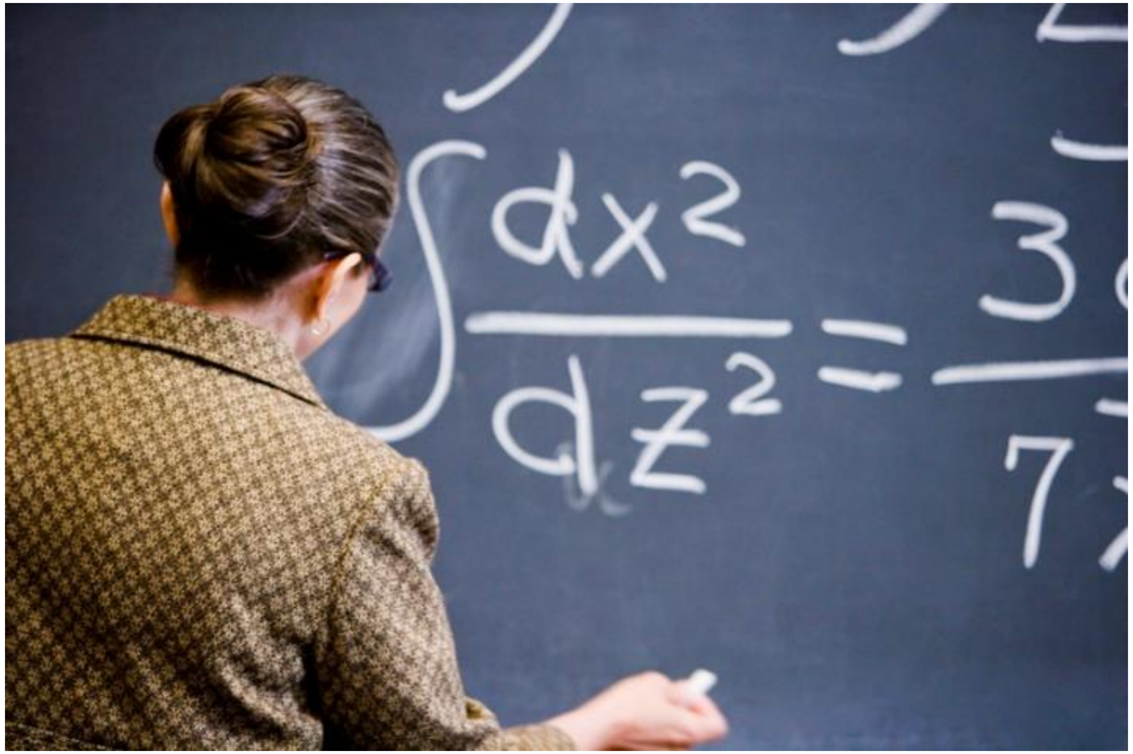
HPE Application Tuner Express (HPE-ATX):

Launching Applications on NUMA Nodes and CPUs

Version. 1.0.3
May 2020

What is HPE-ATX?

- Utility that makes NUMA unaware applications more NUMA aware
  - **No application changes are needed!**
- Controls the distribution of an application's processes and threads in a NUMA environment
  - Several NUMA node and CPU Launch Policies are provided to obtain an optimal distribution
- HPE-ATX vs `numactl`:
  - `numactl` **constrains** an application to a set of NUMA nodes
  - HPE-ATX **distributes** an application around a set of nodes
- Benefit of HPE-ATX varies by platform and application
  - Higher socket count platforms benefit more than lower socket count platforms
  - NUMA-unaware applications benefit more than applications built with NUMA awareness



Hewlett Packard Enterprise

ATX Launch Policies

ATX Process and Thread Launch Policies

– Launch Policies control how an application’s processes and threads are distributed among the NUMA nodes on the system

– There are both process launch policies and thread launch policies.

– The process and thread policies do not have to be the same type in an application.

- Process launch policies govern the NUMA affinity of a child process created from `fork()`, `vfork()`, etc.
- Thread launch policies govern the NUMA affinity of a sibling thread created from `pthread_create()`, `clone()`, etc.

– Launch policies are inherited by a process or thread from it’s creator.

– Launch policies are also inherited across an exec call when starting a new executable image.

ATX Launch Policy types

There are 6 basic launch policy types (available for both processes and threads):

– **Round-Robin (RR):** Each time a process (or thread) is created it will be launched on the next NUMA node in the list of available nodes. This ensures even distribution across all nodes

– **Fill-First (FF):** Each time a process (or thread) is created it will be launched the same NUMA node until we have created the same number of processes (or threads) in the node as there are CPUs in that node (<ncpu>). Once <ncpu> processes have been created future creation will take place in the next NUMA node.

– **Packed (Pack):** All processes (or threads) will be launched on the same NUMA node

– **RR-Packed:** Only the direct children of the specified command will be launched among the available numa nodes in a round-robin manner. Any processes created by the direct children will inherit their launch node from their creator but will not implement process launch policies (essentially all processes created by a direct child of the specified command will be packed into the same numa node).

– **Memfree:** Processes (and threads) will be launched in a round-robin manner until free memory in each numa node is less than a specified percentage, after which processes (and threads) will be launched on the numa node containing the most free memory.

– **None:** No launch policy is defined. Any child process or sibling thread that is created will inherit any NUMA affinity constraints from it’s creator.

Hewlett Packard
Enterprise

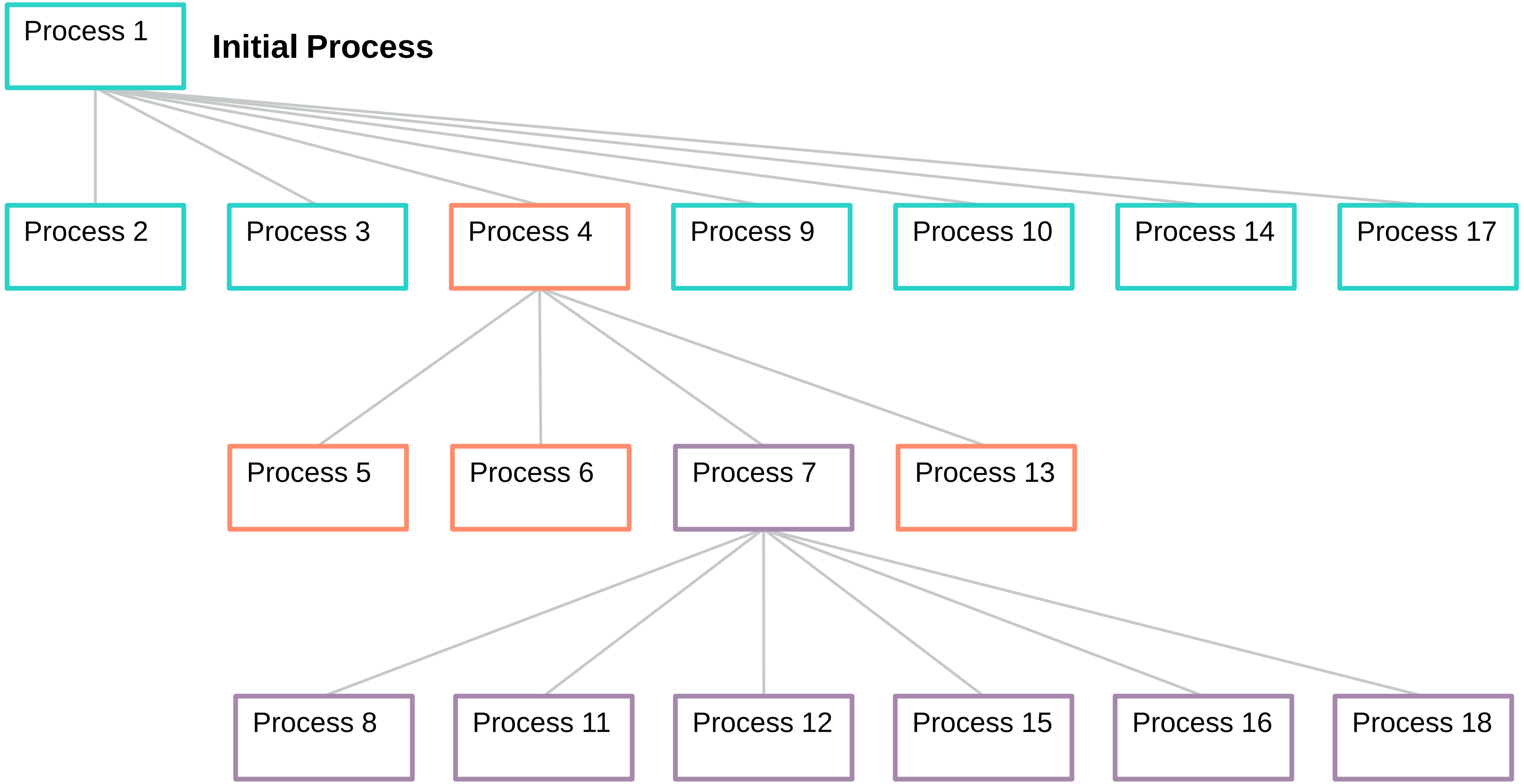
Process Launch Trees

- The process launch tree is the set of processes that the process launch policy should be applied to.
- The initial process started by ATX (as specified by the user) forms the root of a process launch tree.
- **Tree-based policies:**
  - All processes created by the initial process or any of it’s descendants, regardless of how deep the parent/child tree is, are in the same launch tree.
  - All processes will be launched relative to one another in the order they are created according to the launch policy specified.
- **Flat-based policies:**
  - The initial process and only its direct children form a launch tree. The initial process and it’s direct children will be launched relative to one another in the order they are created according to the launch policy specified.

    If one of the children, e.g. child A, creates another process then child A becomes the root of a new process launch tree. All of the direct children of child A will be launched relative to one another in the order they are created according to the launch policy specified.
- **Packed-based policies:**
  - All processes and their children will be packed into one node, regardless of whether that node becomes overloaded.

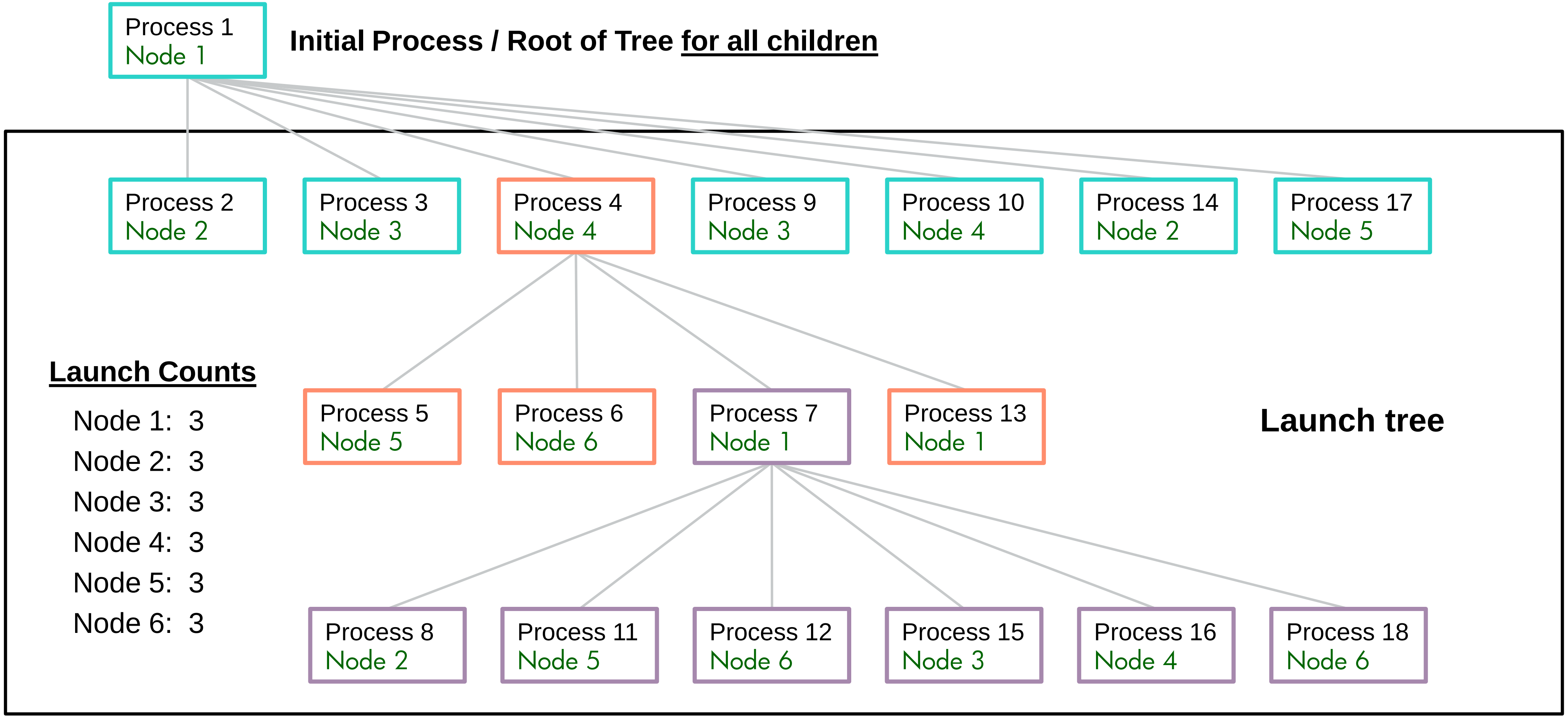
Launch Policy: Sample parent/child tree

Processes are created in the order specified (the process number in each box)



Launch Policy: Process RR-Tree example

Nodes 1-6 available. Processes are created in the order specified. Node # is the resulting node affinity



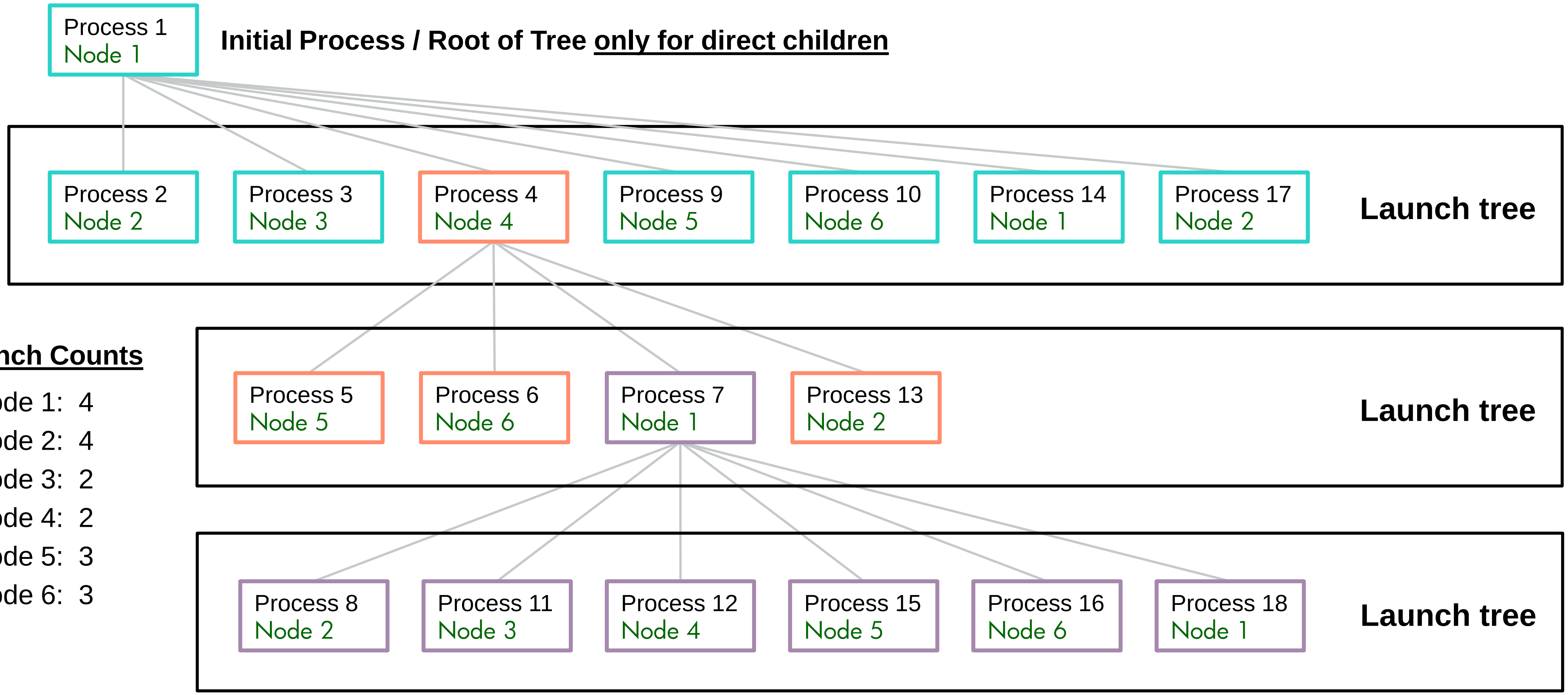
Launch Counts

Node 1:	3
Node 2:	3
Node 3:	3
Node 4:	3
Node 5:	3
Node 6:	3

Hewlett Packard
Enterprise

Launch Policy: Process RR-Flat example

Nodes 1-6 available. Processes are created in the order specified. Node # is the resulting node affinity



Initial Process / Root of Tree only for direct children

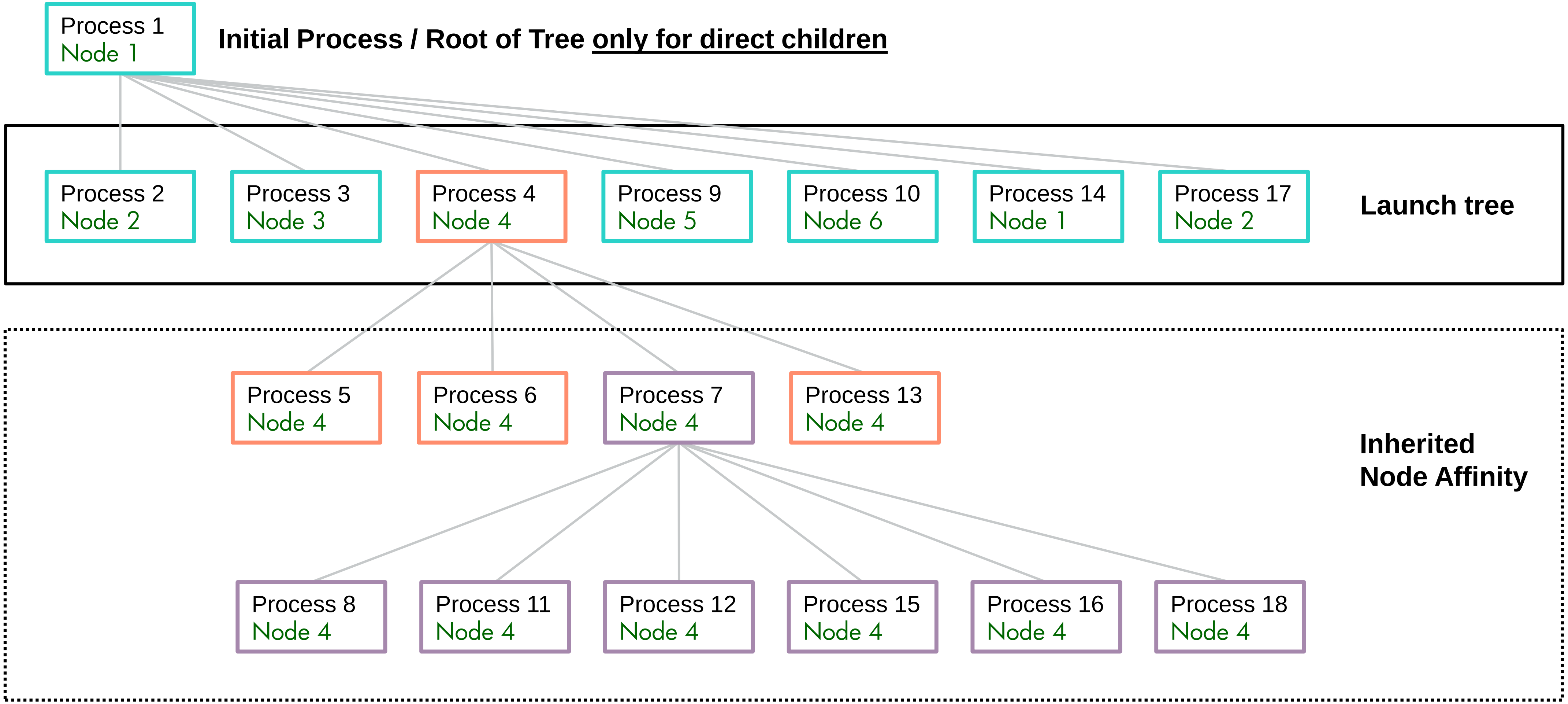
Launch Counts

Node 1:	4
Node 2:	4
Node 3:	2
Node 4:	2
Node 5:	3
Node 6:	3

Hewlett Packard
Enterprise

Launch Policy: Process RR-PACK example

Nodes 1-6 available. Processes are created in the order specified. Node # is the resulting node affinity



Hewlett Packard Enterprise

Launch Policies and the Initial Thread

- Single threaded, or non-threaded, processes still have a single thread (of control) – the initial/main thread.

- When a process is created it is governed by the process launch policy. This policy is applied to the initial/main thread of the process.

- When the process creates it's first thread (this is really the second thread since the initial/main thread of the process was the first thread) the thread launch policy of the process takes effect for thread creation.

- However, the initial/main thread already has already been assigned a launch node according to the process launch policy. This launch node is not changed.
  - Instead, this node is used as the starting node to implement the specified thread launch policy for all threads created by this process.
  - If no thread launch policy was specified all created threads will inherit the launch node from their process.
    - If a process policy is specified, but no thread policy is specified, the thread policy essentially defaults to the **pack** policy due to the inheritance rules.

Thread Launch Policies: Tree vs Flat

– A tree-based thread policy applies to all threads created by the initial process and any of its descendants, regardless of how deep the parent/child tree is (the same as process tree policies).

  - All threads in the process launch tree are launched relative to one another according to the specified thread launch policy.

– A flat-based thread policy applies only to the threads inside a single process.

  - All threads of a single process are launched relative to one another according to the specified thread launch policy.
  - Each process manages its own flat-based thread launch tree.

CPU Launch Option

- ATX also has a CPU Launch option
- You must specify a process or thread NUMA launch policy when using the CPU launch option
- ATX will first determine the launch node according to the specified process or thread NUMA launch policy
- Once the launch node has been selected ATX will select a specific CPU within that node to launch the process or thread
  - CPU selection is round-robin within the node.

ATX Command Line Options

ATX Semantics

```
hpe-atx  [ options ]  [--]  command  [ command arguments ... ]
Options are:
         -p <policy>, --process=<policy>
                 Set the process launch policy to <policy>.
                 <policy> can be one of:
                         rr_tree  rr_flat  ff_tree  ff_flat  rr_pack  memfree_tree  memfree_flat  pack  none

         -t <policy>, --thread=<policy>
                 Set the thread launch policy to <policy>.
                 <policy> can be one of:
                         rr_tree  rr_flat  ff_tree  ff_flat  memfree_tree  memfree_flat  pack  none

         -c, --cpu
                 Also apply the CPU launch policy after determining the
                 launch node based on the process or thread launch policy

         -n <node-list>, --nodes=<node-list>
                 Only use the nodes specified in <node-list> which is
                 in the format of "2-4", "1,3,5", or "1,3,6-8“ (same format as numactl)
                 (the default is the current node affinity)

         -m <limit>, --memfree=<limit>
                 limit is a number from 0 to 100 representing the percent of free memory when the memfree policy
                 should switch from rr to memfree  (the default is 50)

         -l <log-file>, --log=<log-file>
                 Enable logging of process and thread creation (the default is no logging)

         -e <error-file>,   --error=<error-file>
                 Write any errors encountered to this file also

         -r, --remove-data-files
                 Remove any old leftover data files from previous invocations of hpe-atx

         -w, --write-by-other
                 Log and data files will be created with writer permission by “others”
```

ATX Semantics – License Options

```
hpe-atx  [ options ]
Options are:
        -i, --license-instant-on
               Install the free 60-day trial license. This operation is
               performed automatically when hpe-atx is installed.
               This option cannot be used with other options.

        -s, --license-status
               Display information on the currently installed valid hpe-atx license.
               This option cannot be used with other options.

        -S, --license-status-all
               Display information on all currently installed valid hpe-atx licenses.
               This option cannot be used with other options.

        -a <license-file>, --license-add=<license-file>
               Apply the license specified in <license-file> to hpe-atx.  This option
               cannot be used with other options.  This option requires system
               administrator (root) privileges.
```

Initial Launch Nodes Option: `-n` and `--nodes`

– By default ATX will use all nodes available from its node affinity when it starts.

- ATX will adhere to any previous NUMA node affinity through `numactl` or cpusets.

– The application to launch may be constrained to a smaller set of NUMA nodes by using the `-n <node-list>` or `--nodes=<node-list>` option.

- For example, on a 16-processor system the application may need to be constrained to 8-processors

– `<node-list>` has the same format as a node list for the `numactl` command:

- `"1"` or `"1,2,3"` or `"1-3"` or `"1,3-5"` or `"1-3,5-7“` range formats may be used.
- A relative `node-list` may be specified as +1,2,3 or +1-2 and so forth. The `+` indicates that the node numbers are relative to the process' set of allowed nodes.
- An inverse `node-list` can be specified as !1-3 or !1,2,3 and so forth. The `!` indicates that all allowed nodes in the current cpuset except these nodes should be used.
- If the keyword `all` is specified rather than a list of nodes it means use all the nodes in the current cpuset.

Hewlett Packard
Enterprise

Log File Option: `-l` and `--log`

- A `log-file` can be specified to record launch events by using the `-l <log-file>` or `--log=<log-file>` option.
- The following events are logged:
  - Process creation – logged by the parent process
  - Process startup – logged by the child process
  - Process exec – one of the exec* APIs was called to start a new executable image
  - Normal process termination through the exit APIs (implicitly or explicitly)
  - Thread creation – logged by the creating thread
  - Thread creation – logged by the newly created thread
- Abnormal process termination events (e.g., killed by a signal) are not logged
- Thread termination events are not logged

Log File Sample

```
Timestamp  | Entry# |    TID |    PID |   PPID | Node | CPU | Log Message           | cmdline
  0.000155 |      1 |  48036 |  48036 |  47702 |    1 |  76 | initial exec start    | tst/sjn_test
  0.000680 |      2 |  48036 |  48036 |  47702 |    1 |  76 | Created PID 48039     | tst/sjn_test
  0.000902 |      3 |  48039 |  48039 |  48036 |    2 |  30 | child start in fork() | tst/sjn_test
  0.001131 |      4 |  48039 |  48039 |  48036 |    2 |  30 | exit()                | tst/sjn_test
  1.001081 |      5 |  48036 |  48036 |  47702 |    1 |  76 | Created PID 48040     | tst/sjn_test
  1.001306 |      6 |  48040 |  48040 |  48036 |    3 |  45 | child start in fork() | tst/sjn_test
  1.001539 |      7 |  48040 |  48040 |  48036 |    3 |  45 | _exit()               | tst/sjn_test
  2.001390 |      8 |  48036 |  48036 |  47702 |    1 |  76 | Created PID 48041     | tst/sjn_test
  2.001586 |      9 |  48041 |  48041 |  48036 |    1 |  77 | child start in fork() | tst/sjn_test
  2.001804 |     10 |  48041 |  48041 |  48036 |    1 |  77 | _Exit()               | tst/sjn_test
  3.001791 |     11 |  48036 |  48036 |  47702 |    1 |  76 | Created PID 48042     | tst/sjn_test
  3.001996 |     12 |  48042 |  48042 |  48036 |    2 |  31 | child start in fork() | tst/sjn_test
  3.002241 |     13 |  48042 |  48042 |  48036 |    2 |  31 | _exit()               | tst/sjn_test
  4.002100 |     14 |  48036 |  48036 |  47702 |    1 |  76 | Created PID 48043     | tst/sjn_test
  4.002300 |     15 |  48043 |  48043 |  48036 |    3 |  46 | child start in fork() | tst/sjn_test
  4.002519 |     16 |  48043 |  48043 |  48036 |    3 |  46 | _exit()               | tst/sjn_test
  5.002491 |     17 |  48036 |  48036 |  47702 |    1 |  76 | Created PID 48044     | tst/sjn_test
  5.002683 |     18 |  48044 |  48044 |  48036 |    1 |  78 | child start in fork() | tst/sjn_test
  5.002903 |     19 |  48044 |  48044 |  48036 |    1 |  78 | _exit()               | tst/sjn_test
```

Hewlett Packard
Enterprise

Memfree Option: `-m` and `--memfree`

- The memory based launch policies additionally provide the `-m <limit>` or `--memfree=<limit>` option to control when the launch policy should switch from round-robin to free memory based launching.
- `<limit>` must be a value from `0` to `100`. The default value is `50`. This value represents a free memory percentage.
- Processes/threads will be initially be launched in a round-robin manner on the available nodes.
- When the amount of free physical memory on a node is less than the `<limit>` percentage threshold that node will be removed from the available list of nodes on which new processes and threads can be launched.
  - If the removed node later has free memory greater than the `<limit>` percentage threshold the node will be added back to the list of nodes on which new processes/threads can be launched.
- If/when the amount of free physical memory on all nodes is less than the `<limit>` percentage threshold new processes/threads will be launched on the node that has the most free physical memory.
- Setting `<limit>` to `0` results in a pure round-robin launch policy.
- Setting `<limit>` to `100` results in a pure free memory based launch policy.

Memfree Option: `–m` and `--memfree`

- **NOTE:**
  - Physical memory is not allocated until a process/thread both allocates memory and initializes that memory.
  - Depending on how an application and it’s processes/threads are started this policy may not provide the desired effect of launching based on free physical memory.
  - If previously launched processes/threads have not yet allocated and initialized all of their memory future process/thread launch decisions will be made with incomplete per node free memory statistics.
  - Free memory based launch policies will not be effective when many processes/threads are created simultaneously.

Remove Data Files Option: `-r` and `--remove-data-files`

– ATX internally maintains an internal data file that is used to keep track of which Node/CPU should be used for the next process or thread launch.

– It is possible that ATX could leave old unused data files around.

– By default, when ATX starts it looks for and removes any old unused internal data files from a previous invocation of ATX that is no longer running.

  – While these data files do not claim a lot of space you can use the `-r` or `--remove-data-files` option by itself to remove any old unused internal data files.

  – For example:

    `hpe-atx -r` or `hpe-atx --remove-data-files`

– This option does not provide output.

– If unused data files are found they are silently removed.

Write-by-Other Option: `-w` and `--write-by-other`

- By default ATX will create the `<log-file>` as well as an internally maintained data file with read/write permissions for owner / group and read permission for other.
  - i.e., 0664 or `-rw-rw-r--`
- Some applications may change their user or group ID causing access permission problems for the *`<log-file>`* and the internal data file.
- The `-w` or `--write-by-other` option will cause the *`<log-file>`* and the internal data file to be created with read/write permissions for owner, group, and other.
  - i.e., 0666 or `-rw-rw-rw-`
- Note: the file-creation mode mask inherited from the parent process or set with the shell's `umask` command may cause files to be created with more restrictive permissions.
  - To ensure ATX creates files with the permissions specified above set the umask with the command:

```
umask 0
```

Error File Option: `-e` and `--error`

- If any errors are encountered by ATX the error will be written to `stderr`.
  - Run-time errors (not process startup errors) will also be written to the `log-file` if logging was specified.
- Some applications close `stdout` and `stderr` which can make error reporting difficult.
- The `-e <error-file>` or `--error=<error-file>` option will cause ATX to additionally write any error it encounters to the file `error-file`.
- This file will only be created if ATX encounters an error launching the specified command or any of its children.
- It is recommended to always use this option unless you know for certain that the specified command does not close `stdout` and `stderr`.

ATX Command Line Examples

Examples

– **hpe-atx -p rr_tree -- app args**

- Run `app` with arguments `args` using the `rr_tree` process launch policy

– **hpe-atx -p rr_tree -t ff_flat app args**

- Run `app` with arguments `args` using the `rr_tree` process launch policy and the `ff_flat` thread launch policy

– **hpe-atx -p rr_flat -c app args**

- Run `app` with arguments `args` using the `rr_flat` process launch policy as well as a CPU launching within the node

– **hpe-atx -p ff_tree -l log.txt app args**

- Run `app` with arguments `args` using the `rr_tree` process launch policy and log launch operations to the file `log.txt.`

– **hpe-atx -p ff_flat -n 1-3 app args**

- Run `app` with arguments `args` using the `ff_flat` process launch policy and use nodes `1-3` to launch the process and threads created by `app.`

– **hpe-atx -p rr_flat -e errors.txt app args**

- Run `app` with arguments `args` using the `rr_flat` process launch policy. Log any errors encountered by `hpe-atx` to the file `errors.txt`.

Hewlett Packard
Enterprise

ATX Error Handling and Constraints

Error Handling

– If ATX encounters an error while launching processes and threads it has three ways of handling the error:

1) Errors that occur before or during the launch of the specified command are treated as fatal startup errors and ATX will terminate. These type of errors are usually errors related to bad parameters being passed to ATX.

2) Some errors are survivable errors that do not need to stop ATX or the specified command from continuing. An example of a survivable error is if writing to the log file fails then logging is disabled, however, the specified command and it's children will continue to run.

3) After the specified command has started, if ATX encounters a fatal error it will treat the error as a disable error. ATX will disable itself in the process that encountered the error. All of the initial launch nodes will be used for that process as well as any processes or threads it creates.

– This approach for surviving with reduced functionality and/or disabling ATX when an error is encountered allows the user to cleanly shutdown the specified command at an opportune time (as opposed to terminating the specified command).

Constraints

– ATX is not currently supported for use within a virtual guest environment

– ATX does not support `32-bit`, `setuid` or `setgid` executables

- If encountered ATX will disable itself for that executable and any processes or threads it creates.
- The executable (and all of it's children) will inherit the launch node or cpu from its creating process
- The executable is still allowed to run
- If a *`<log-file>`* was specified a log entry will be created to notify that this type of executable was encountered

– ATX does not recognize and reconfigure itself as cpus are placed in an offline state.

- If encountered ATX will disable the launch features in that process as well as any processes or threads it creates.
- The process will be allowed to use all of the initial launch nodes
- The process will continue to run

– If a cpu is placed in an online state after starting ATX, and the cpu launch option was specified, the new cpu brought online will not be used by ATX.

– If an entirely new NUMA node of cpus is placed in an online state after starting ATX the new node brought online will not be used by ATX.

ATX Tuning Thoughts

Quick Tuning thoughts…

- Run `hpe-atx -l my_log.txt <cmd>` (no launch policy specified) and look at the resulting log file to understand the entire process tree, parent/child relationships, creation order, etc.
    - Are there only processes created? If yes, there is no need to try the `-t` thread policies.
    - Are there only threads created? If yes, there is no need to try the `-p` thread policies.
    - Is there only one level of child processes? If yes, there is no need to try the `rr_tree` or `ff_tree` process policies.
  - Answering questions like these will help arrow down which ATX options to try.
- For applications where there is only one executable to launch it’s pretty simple – just try the different launch policies (with and without CPU launching).
- For applications where there are multiple executables to be started you might want to look at each executable separately.
  - It might be that each executable needs a different launch policy and should be launched by separate ATX invocations
  - It might be that only some executables in an application need an ATX launch policy and others started normally

Quick Tuning thoughts…

- Run `hpe-atx -l my_log.txt <cmd>` (no launch policy specified) and look at the resulting log file to understand the entire process tree, parent/child relationships, creation order, etc.
    - Are there only processes? Only threads?
    - Is there only one level of child processes? Or multiple levels?
  - Answers to these questions will help arrow down which ATX options to try (`-p` vs `-t` options, tree s flat policies, etc)
- For applications where there is only one executable to launch it's pretty simple – just try all the launch policies (with and without CPU launching).
- For applications where there are multiple executables to be started you might want to look at each executable separately.
  - It might be that each executable needs a different launch policy and should be launched by separate ATX invocations
  - It might be that only some executables in an application need an ATX launch policy and others started normally

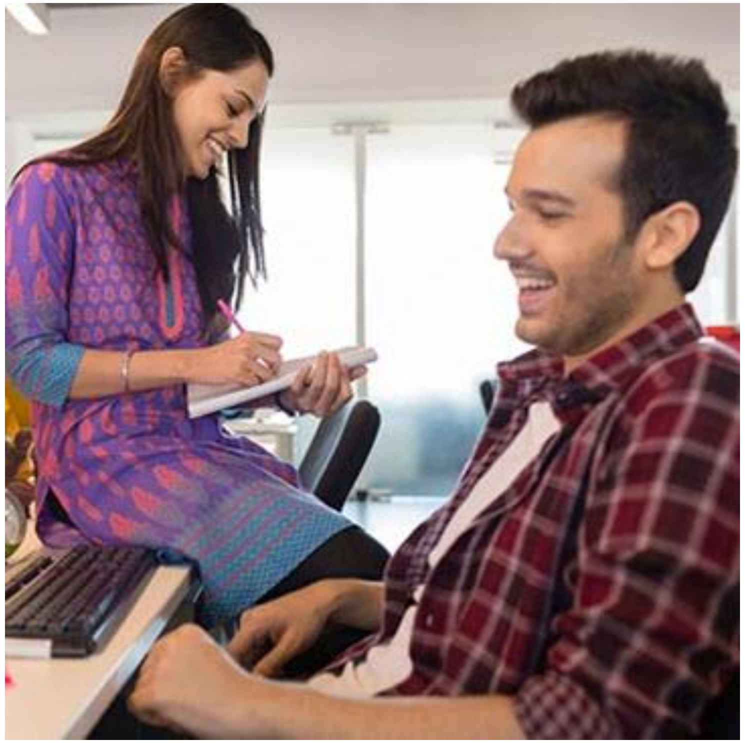
Hewlett Packard Enterprise

Additional Resources

For additional information

Publicly Available Resources

- HPE-ATX product page:
  - http://downloads.linux.hpe.com/SDR/project/hpe-atx
- HPE-ATX download:
  - HPE My License Depot
    - https://myenterpriselicense.hpe.com/cwp-ui/evaluation/HPE-ATX
  - HPE Software Delivery Repository
    - http://downloads.linux.hpe.com/SDR/project/hpe-atx/repo.html
- Documentation: Installing HPE-ATX
  - http://downloads.linux.hpe.com//SDR/project/hpe-atx/Installing_HPE-ATX_v_1.0.3.pdf
- Documentation: Launching applications with HPE-ATX
  - http://downloads.linux.hpe.com//SDR/project/hpe-atx/Using_HPE-ATX_v_1.0.3.pdf
- Documentation: HPE-ATX Performance Proof Points
  - http://downloads.linux.hpe.com//SDR/project/hpe-atx/HPE-ATX_proof_points_v3.pdf
- Demo Videos on the HPE Solution Demonstration Portal
  - https://hpedemoportal.ext.hpe.com/search/HPE-ATX



Hewlett Packard
Enterprise

Thank you