Real Time Linux Scheduling Comparison

Vince Bridgers
Software Architect
Altera Corporation

Who am I?

- Software Developer and Architect at Altera Corporation
  - Open Source Development Activities in Austin, Texas
- Open source projects
  - Linux – LTSI, Real-time and Custom for ARM SOCs
  - UBoot
- Technologies …
  - Altera FPGA IP Enablement
  - Embedded Software and Systems
  - Ethernet, IEEE 1588
  - Automated testing

Agenda

- Introduction to Real Time Linux & LTSI
- Creating a Custom Real Time Linux Kernel
- A Methodology for Comparing Scheduling Latency
- Some interesting results

LTSI and Real-Time Linux

- LTSI Announced in October 2011 at LinuxCon Europe
  - Create a supported Linux kernel for the embedded systems life cycle
  - Industry managed kernel as common ground for the embedded industry
  - Mechanisms for upstreaming activities from embedded systems engineers
- Real Time Linux
  - A set of patches developed over the years to provide soft real time capabilities by allowing pre-emption in the Linux kernel and additional features to improve scheduling determinism.
  - Main Wiki - https://rt.wiki.kernel.org/index.php/Main_Page

Real-Time Classifications

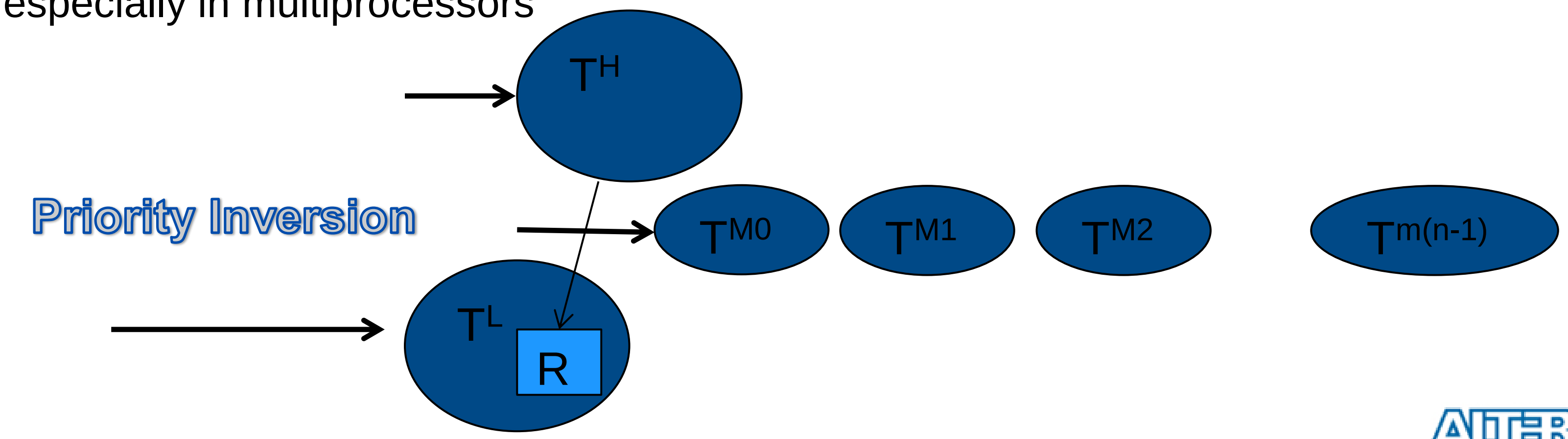
Type of Real Time	Characteristics	Use Cases
Soft Real Time	Subjective Scheduling deadlines, depends on the application	Media rendering on mainstream operating systems, network I/O, flash access
95% Real Time	Real time requirements met 95% of the time, system can compensate 5% of the time.	Voice Communications, data acquisition
100% Real Time	Real time requirements met 100% of the time else manufacturing defects can occur	Factory automation where failure results in manufacturing defects
Safe Real Time	Real time requirements met 100% of the time else serious injury or death can occur	Flight and weapons control, life critical medical equipment

Sources of Non-Deterministic Latency

- Latency is “the interval between stimulus and response”
  - Latin root – latēns : “to lie hidden”
- “Nondeterministic” means the ΔT latency between “stimulus” and “response” falls outside of an accepted upper and lower bound, or cannot be predicted. Known as “Latency Jitter”
- Latency can come from multiple sources ….
  - Unbounded Priority and Interrupt Inversion
  - Scheduling latency (depends on scheduling policies)
  - Interrupt latency
  - Caching and TLB effects – especially in multiprocessors
  - Paging I/O Latency
  - Memory access latency

Scheduling Latency

1) ***ISR***
2) ***Scheduler Invoked***
3) ***Task Picked***
4) ***Context Switch***

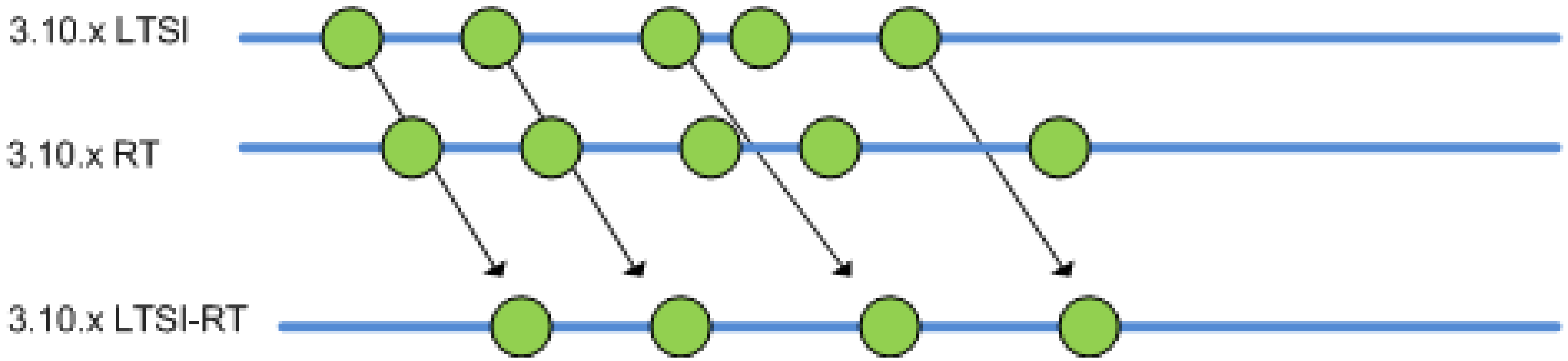


Preempt RT Patch

- Linux RT Preempt is a 95% Real Time System
- RT Preempt Changes …
  - Threaded Interrupts
  - Pre-emptible mutual exclusion (“Sleeping” Spinlocks)
  - Priority Inheritance
  - High Resolution Timer
  - Real time scheduling policies – SCHED_RR and SCHED_FIFO
- “Real Time” applications are expected to make good choices in the application design
  - Make sure commonly used memory is paged in
  - Smart processor and memory management
  - Smart priority assignment and management
- Simply using the RT Preempt patch does not solve all problems. Users must do some work too.
- User must be careful with affinities and priorities

Creating a rebased Linux-RT Kernel

- Checkout the latest 3.10-ltsi kernel
- Checkout the same branch of the Stable Linux RT Kernel
- Rebase …



Creating a Rebased Linux-RT Branch

- A developer can create their own rebased Linux-RT branch from a customized kernel using rebase
- Example steps ….

```
git clone http://git.rocketboards.org/linux-socfpga.git
cd linux-socfpga
git fetch linux-socfpga
git checkout -b socfpga-3.10-ltsi-rt-rebase origin/socfpga-3.10-ltsi
git remote add linux-rt git://git.kernel.org/pub/scm/linux/kernel/git/rt/linux-stable-rt.git
git fetch linux-rt
git checkout –b linux-rt-3.10 linux-rt/v3.10-rt
git checkout socfpga-3.10-ltsi-rt-rebase
git rebase linux-rt-3.10 …
```

- Iterate: Resolve conflicts, git rebase –continue

Building and Testing the Real Time Kernel

- CONFIG_PREEMPT_RT_FULL
- High Resolution Timer
- Make sure power management is off
- Build test …
  - allconfig
  - Allmodconfig
- See online tutorial
  - https://rt.wiki.kernel.org/index.php/RT_PREEMPT_HOWTO

Evaluating Latency

- Comparing averages or max values may not yield interesting results – need comparative statistics to see full potential of latency jitter benefits.
- Measurement Methodology
  - Benchmark uses get time of day as a way to measure request to response latency, multiple block memory read/write threads, multiple ping floods
  - Collect 5000 samples, collect into bins for a histogram
  - Collect “online” statistics for mean, skew, kurtosis, and percentiles
  - Statistics given are accurate to within two decimals points with 95% confidence
- Altera’s Socfpga-3.10-ltsi kernel without RT Preempt patches
- Altera’s Socfpga-3.10-ltsi-rt kernel – Same as above with RT Preempt patches applied
- Measured on Altera’s Cyclone 5 SOC

ALTERA.

Characteristic Workload

- Multiple ping floods – simultaneous transmit and receive network traffic
- Dedicated memory thrashing threads per CPU
  - Large block memory allocation, random reads and writes
- Dedicated threads per CPU uses clock_gettime and clock_nanosleep to cycle threads through process states
- Difference between requested sleep time and measured sleep time is defined to be “scheduling latency” and collected for comparison
- User could create custom workload that’s characteristic of their system design
- ***Disclaimer: This is not intended to be exemplary for all RT use cases!***

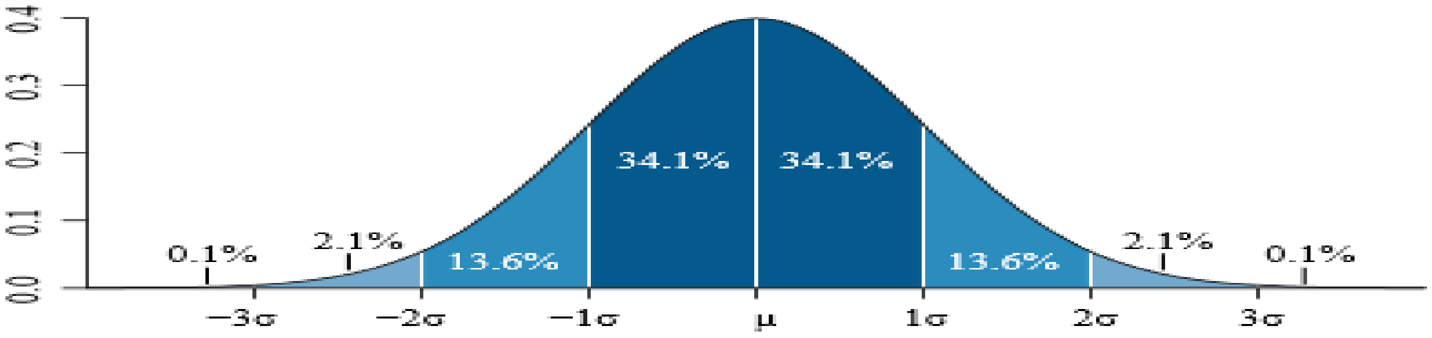
Data Collection Core for Measurements and Comparison

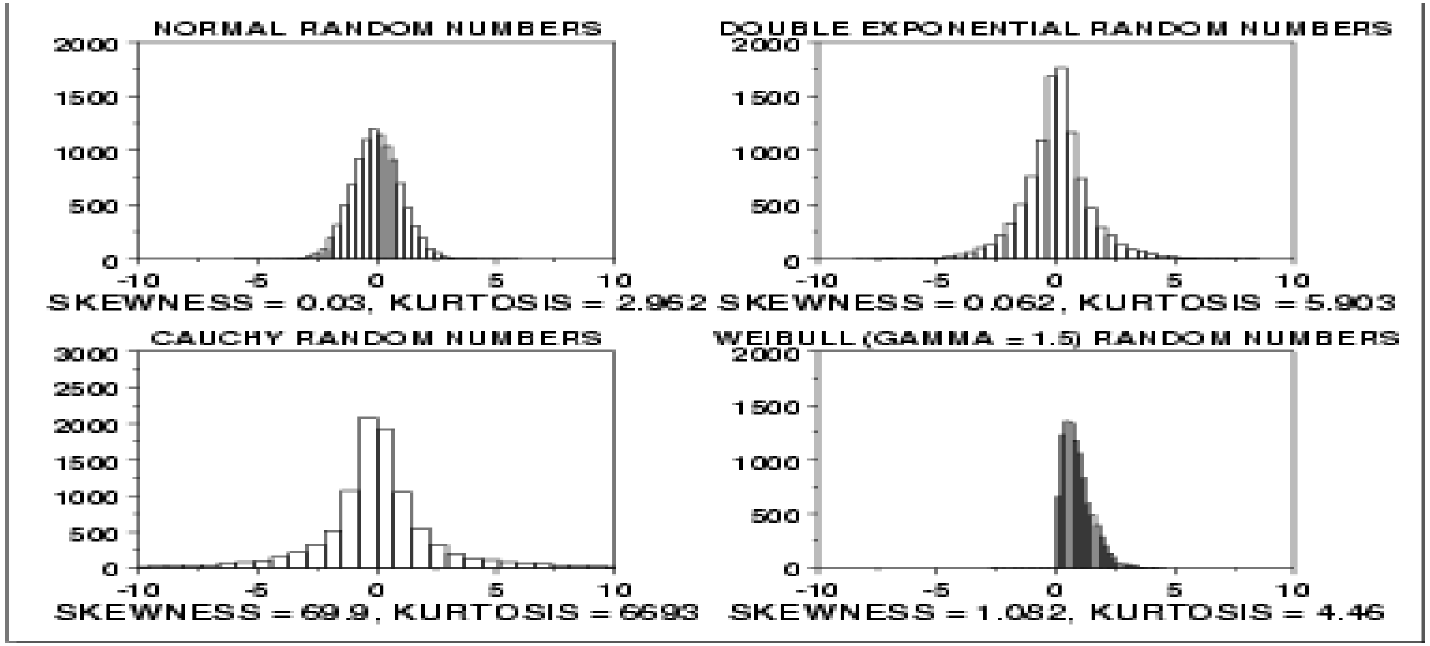
```
ret = clock_gettime(clock[ptctx->clksrc], (&now));
if (ret != 0) {
         fail();
}
req.tv_sec = 0;
req.tv_nsec = 100*(1000*1000);

ret = clock_nanosleep(clock[ptctx->clksrc], 0, &req, NULL);
if (ret != 0) {
         fail();
}
ret = clock_gettime(clock[ptctx->clksrc], (&next));
if (ret != 0) {
         fail();
}
diff = calcdiff(next, now) ;

int delta = (int)(diff-timens(req))/1000;
ptctx->pm_q5->push(delta);
ptctx->pm_q50->push(delta);
ptctx->pm_q99->push(delta);
ptctx->pm_q95->push(delta);
ptctx->pstats->push(delta);
```

Statistics Collection

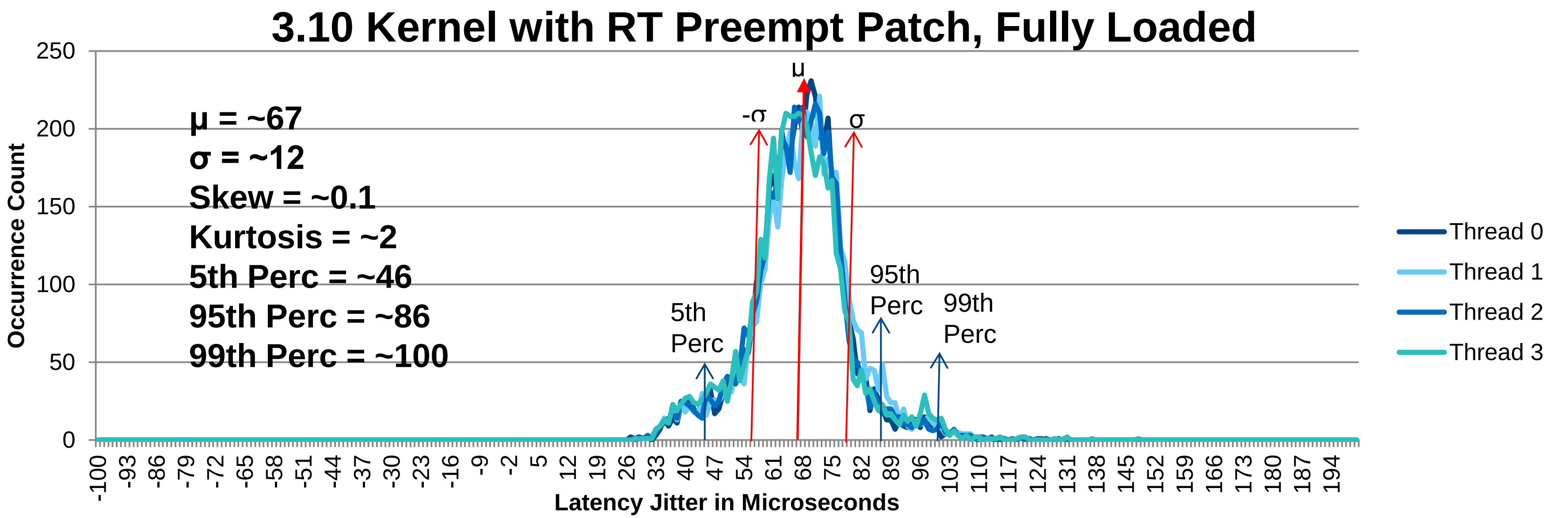
- Percentiles collected "online" using the Piecewise Parabolic Method
- Means, Standard Deviation, and data moment statistics collected in real time using optimized "online" algorithms for collecting statistics
  - See Welford's Algorithm – efficient and numerically stable
  - Methods presents by Timothy Terriberry used to maintain and compute higher order data moments (standard deviation, skew and kurtosis).
- Implemented as a simple, portable, reusable C++ class for applications
- Cumulative and moving averages, standard deviation, skewness, kurtosis, and percentiles.

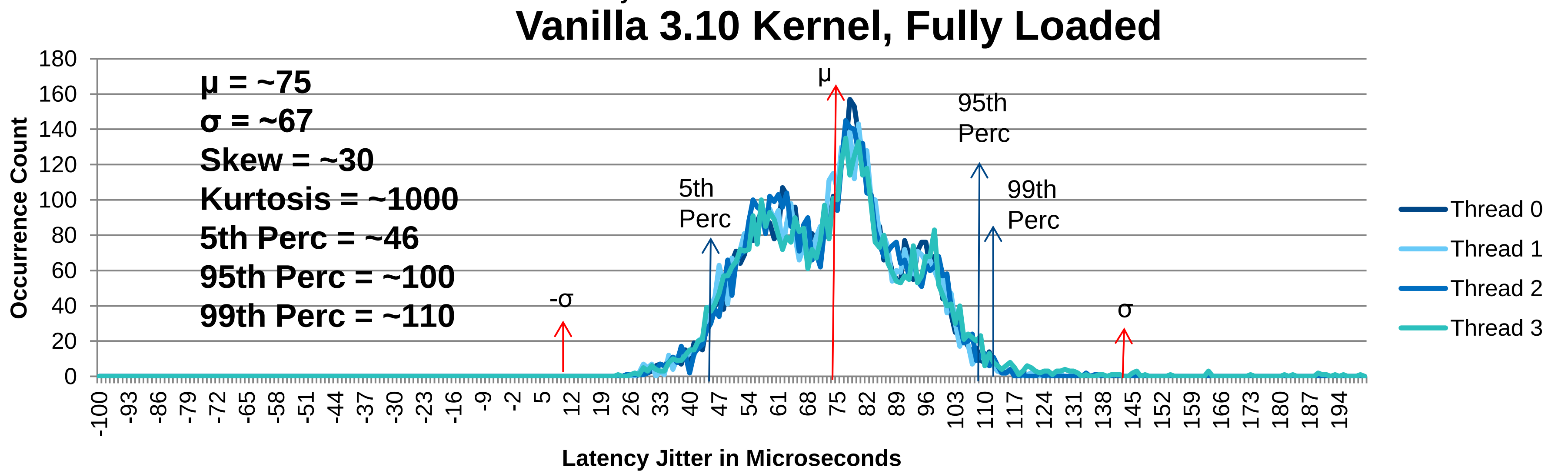
Statistics Review





Scheduling Latency Jitter Comparison





Observations

- Mean comparison shows a clear improvement from vanilla kernel to RT kernel.
- Review of other statistics show that outliers are greatly reduced in RT kernel (skewness and kurtosis).
- Standard deviation is greatly improved in RT kernel
- The 5^{th} percentile is about the same – indicating a “hard” lower bound.

Thank You

References

- LTSI Update : http://lwn.net/Articles/484337/
- Real Time Preemption Overview : http://lwn.net/Articles/146861/
- Altera SOCFPGA LTSI-RT Kernel
  - http://www.rocketboards.org/foswiki/Documentation/AlteraSoCLTSIRTKernel
- Altera GIT Repositories http://rocketboards.org/gitweb/

Welford's Method

- Single pass algorithm – useful for online data.
- A "current" value can be maintained as data samples become available.
- Numerical stability is pretty good
- Computationally efficient
- This algorithm yields mean, standard deviation, and variance.

$$M_1 = 0, S_1 = 0$$
$$M_i = M_{i-1} + \frac{x_i - M_{i-1}}{i}$$
$$S_i = S_{i-1} + (x_i - M_{i-1})(x_i - M_i)$$

Equation 4 - Welford's Method

Higher order moments ….

- Central moments are maintained
- Updated by a "push" operation as samples arrive
- Numerically stable

$$\delta = x - m$$

$$\mu = m' = m + \frac{\delta}{n}$$

$$M_2' = M_2 + \delta^2 \frac{n-1}{n}$$

$$M_3' = M_3 + \delta^3 \frac{(n-1)(n-2)}{n^2} - \frac{3\delta M_2}{n}$$

$$M_4' = M^4 + \frac{\delta^4 (n-1)(n^2 - 3n + 3)}{n^3} + \frac{6\delta^2 M_2}{n^2} - \frac{4\delta M_3}{n}$$

Equation 5 - Central Moments Difference Equations

P2 Method

- Maintains 5 markers on a cumulative distribution curve
- Sample arrives, markers are updated
- Markers correspond to p/2, p, (1+p)/2 and the maximum quantile
- Heights are adjusted using a Piecewise Parabolic (P2) formula.