POWER8 On Chip Controller

Measuring and Managing Power Consumption

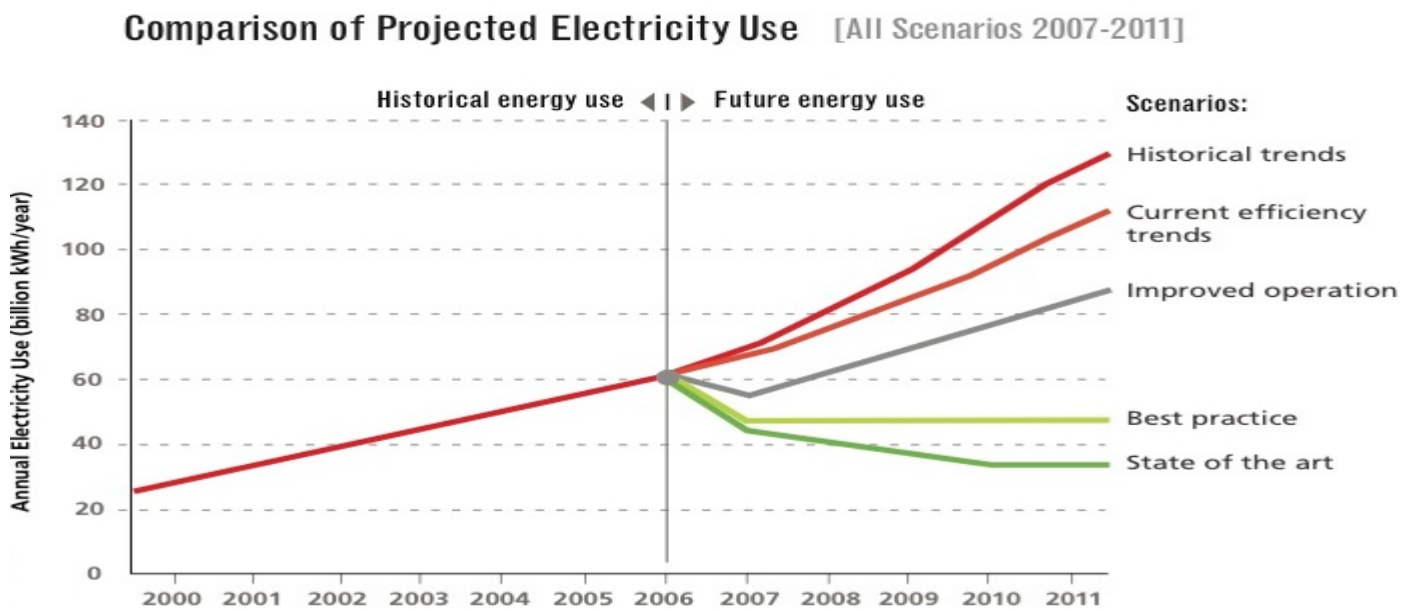
Todd Rosedahl
Charles Lefurgy

Agenda

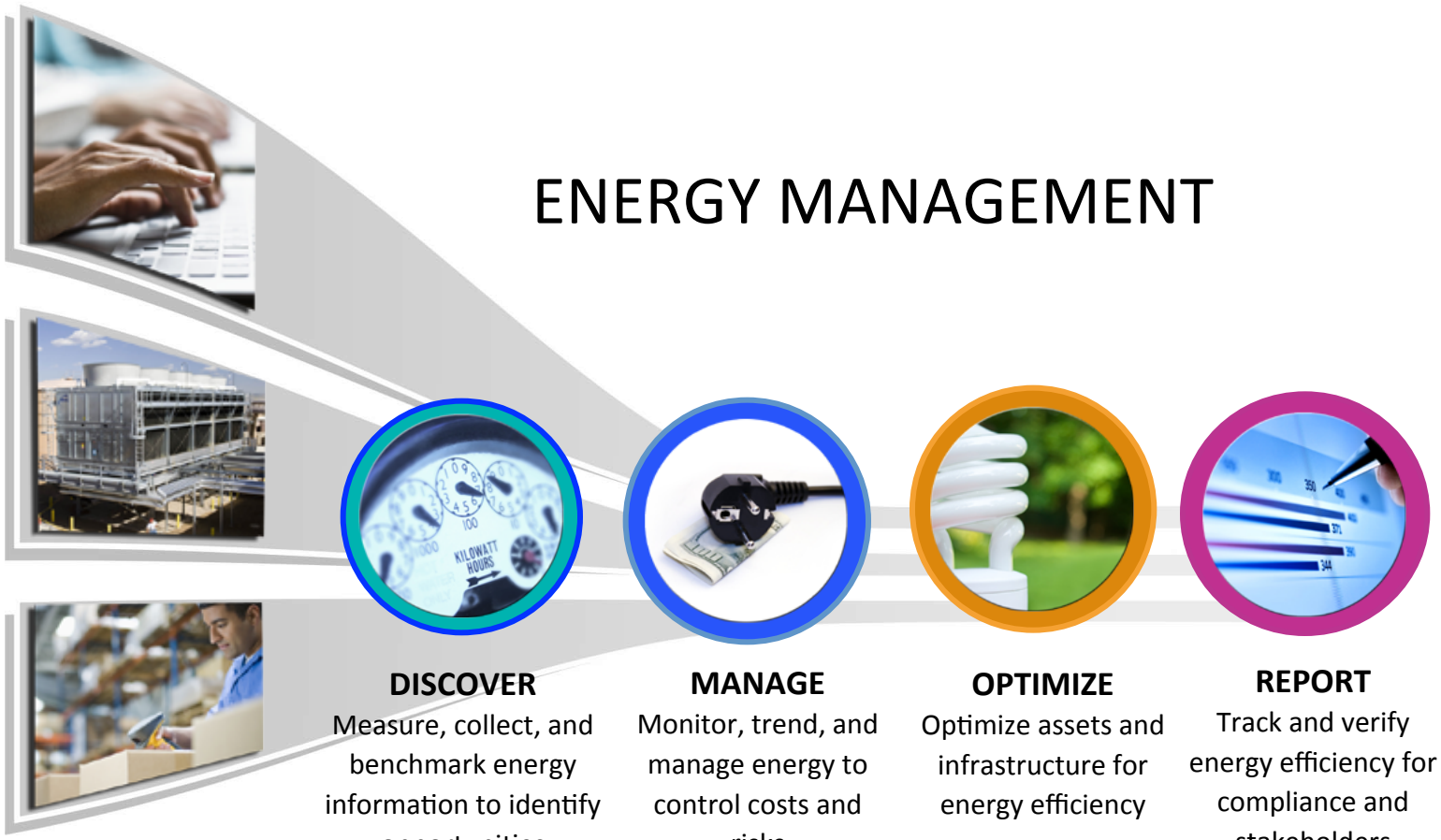
- Introduction/Motivation
- Hardware Block Diagram
- OCC Overview
- Firmware Details
- Power/Thermal Management
- Power/Thermal Measurement
- AMESTER
- Next Steps
- References

Motivation

- US datacenter energy consumption – 91Billion KWH
- 34 500MW power plants
- State of Minnesota – 68Billion KWH
- Power Usage Effectiveness
- Energy Star, SERT, Regulations



A **practical approach** to energy management



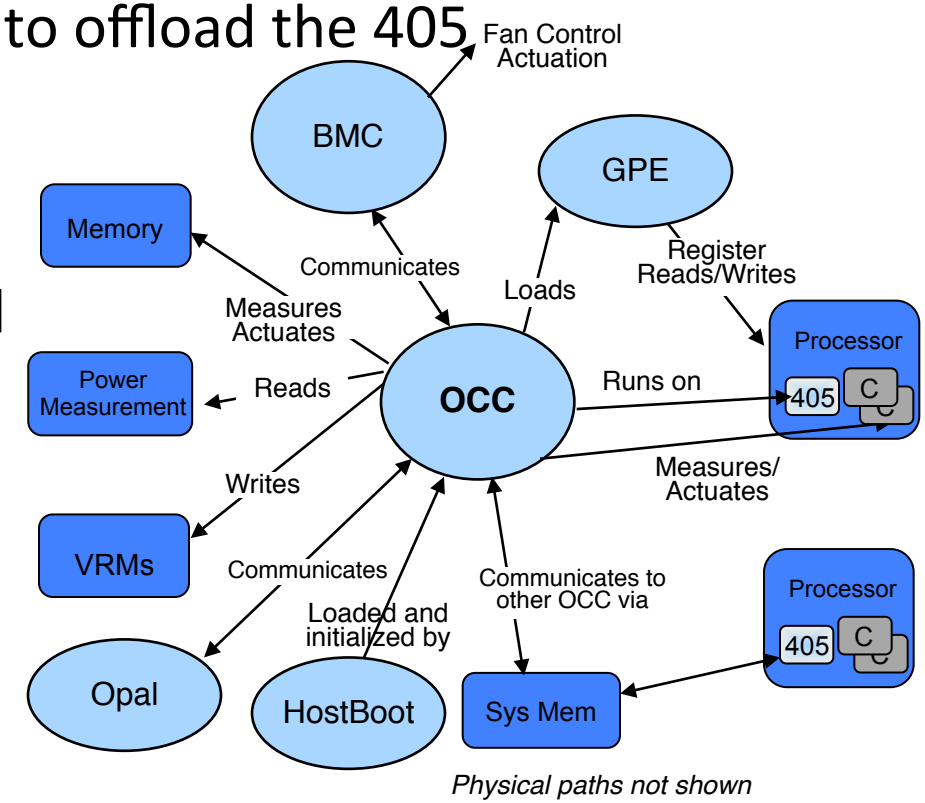
OCC Overview

What is OCC?

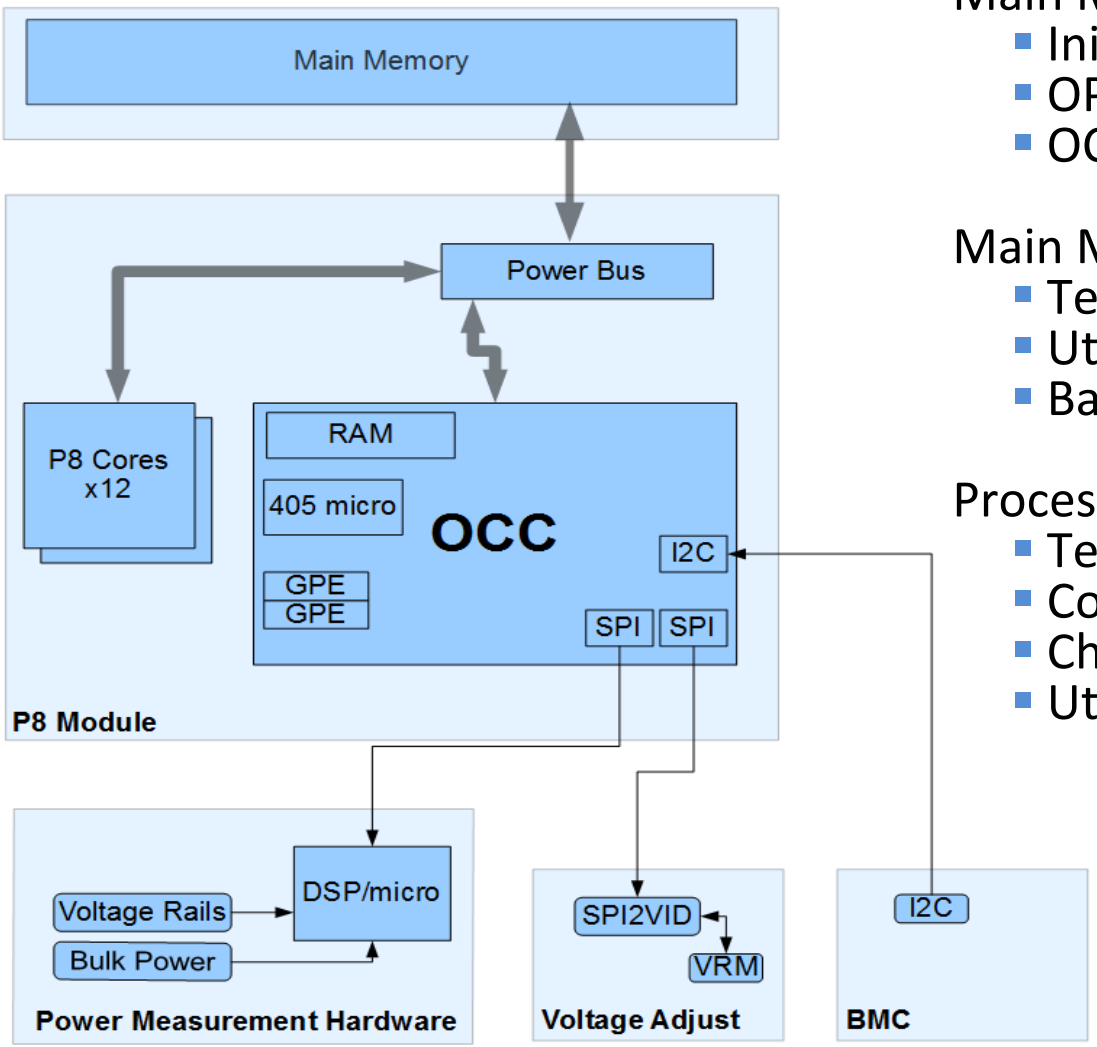
- Hardware/Firmware that controls power, performance & thermal
- 405 processor with 512k dedicated RAM
- General Purpose Engines (GPE) to offload the 405

What does OCC do?

- Reads/controls system power
- Reads/controls chip temps
  - Enables efficient fan control
  - Provides OT protection
- Power Capping
- Fault Tolerance
- Energy saving
- Performance boost



OCC - Hardware Block Diagram



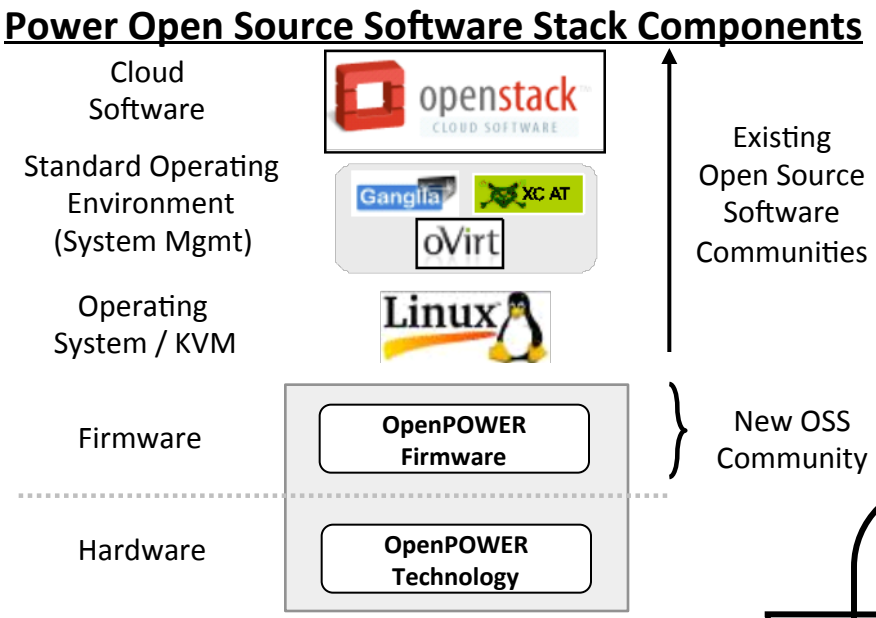
Main Memory Uses
- Initial Load
- OPAL communication
- OCC communication

Main Memory Measurement
- Temperature Sensing
- Utilization Measurement
- Bandwidth control

Processor
- Temperature Sensing
- Core Frequency Control - PSTATES
- Chip Voltage Control – PSTATES
- Utilization Measurement

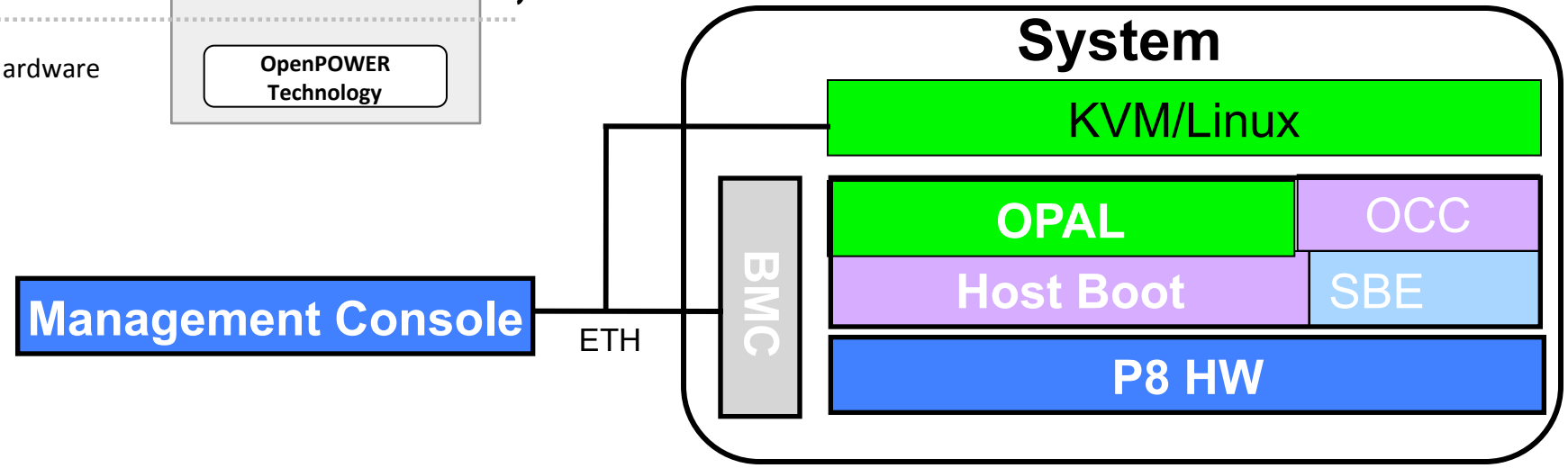
BMC
- Report power/temperature
- Provide Power Cap
- DCMI compliance

Ecosystem Enablement

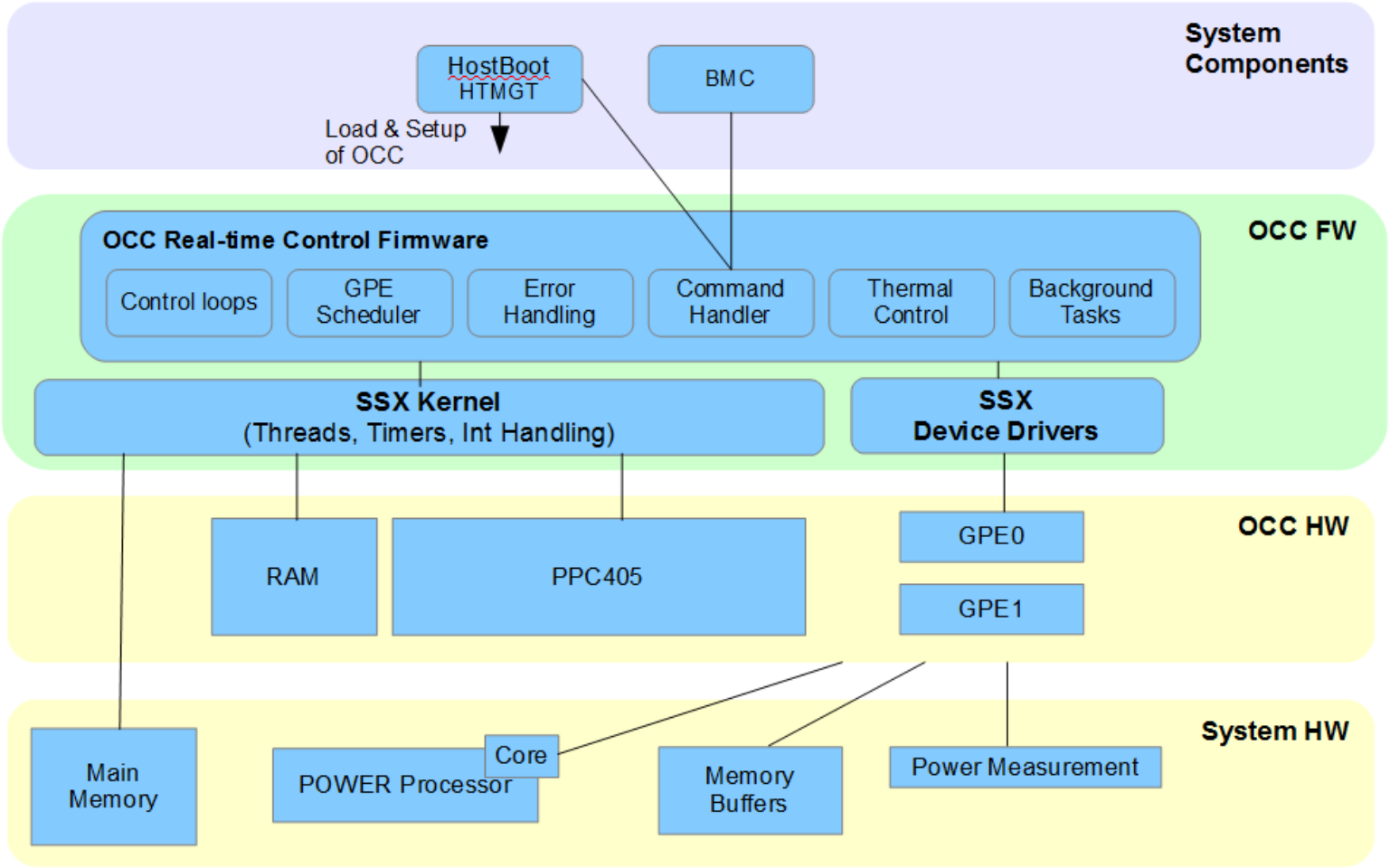


July 2014 Power8 open source firmware stack contributed thru GitHub

Toolkits and resources for porting and optimizing, growing repository on website



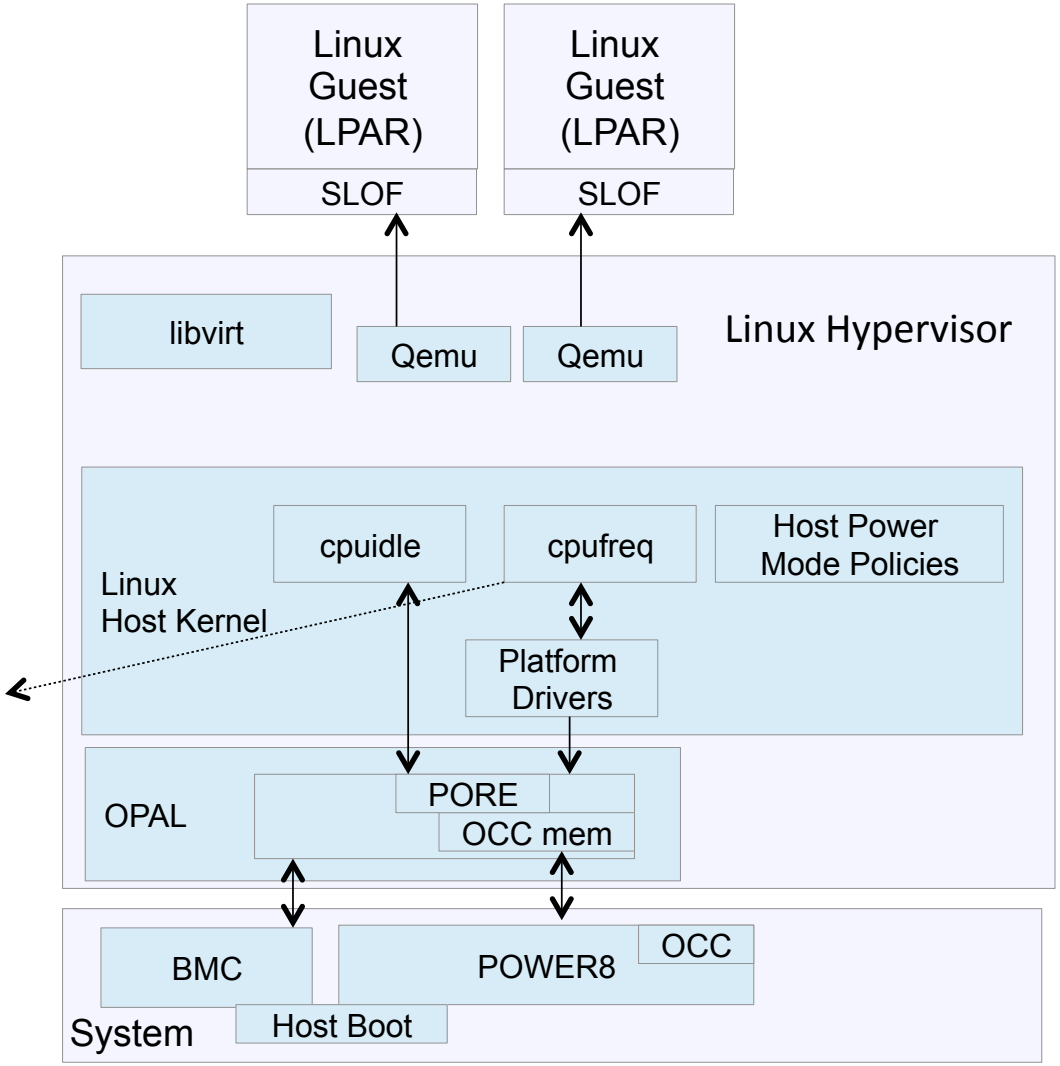
OCC Internal Details



PSTATE Details

Utilization and governor policy are inputs to cpufreq in Hypervisor

Cpufreq – exploits SPR based P-state change



Management Options

Linux Governors – OCC measures/clips for capping/OT

- Balanced
- Performance
- Power save

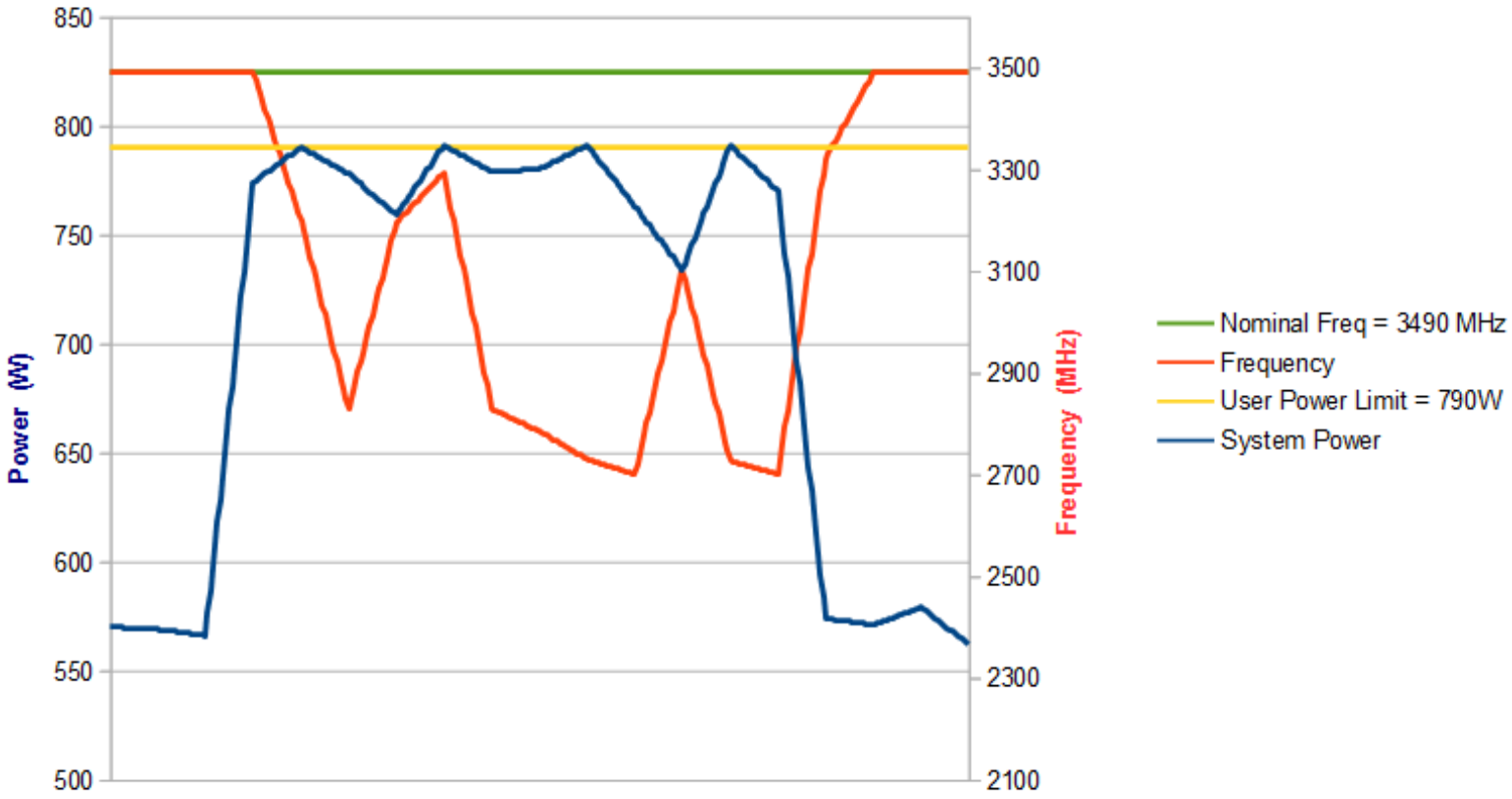
OCC Controls Frequency

- Dynamic Power Save
  - Favor Power or Favor Performance
- Fixed Frequency Mode
- Fixed Maximum Frequency
- Idle Power Save
- Power Capping
- Over Temperature Protection

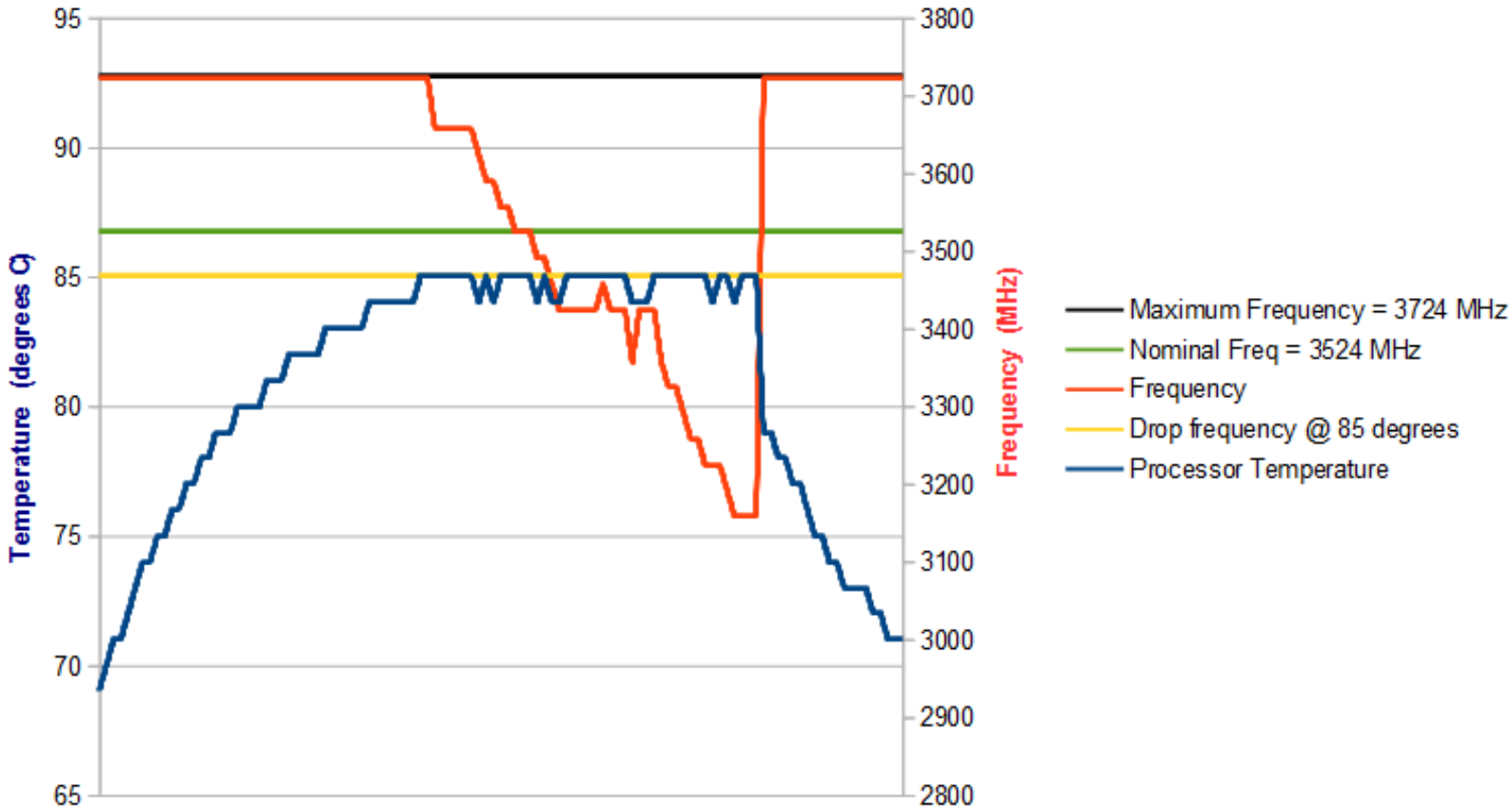
Processor Idle States

- Nap
- Sleep
- Winkle

User Set Power Limit

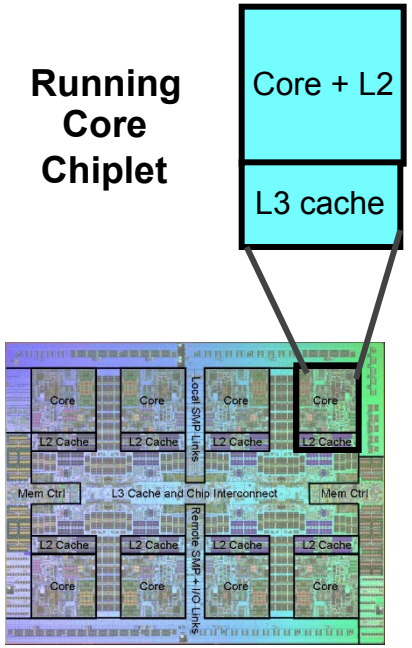


Processor Over Temperature



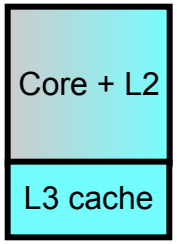
Turn Processors Off – Nap/Sleep/Winkle

Running Core Chiplet



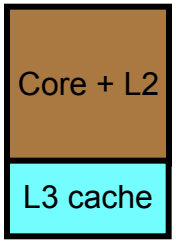
Power7 Chip (8-core)

NAP
(per core)



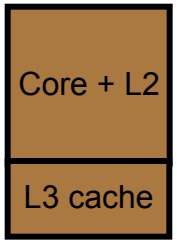
Used by default for short Idle Periods
Stop clocks to only processor core execution engines.
Leave all caches running.
Saves ~ 10% power with ~ X**u**s latency (i.e. fast)

Sleep



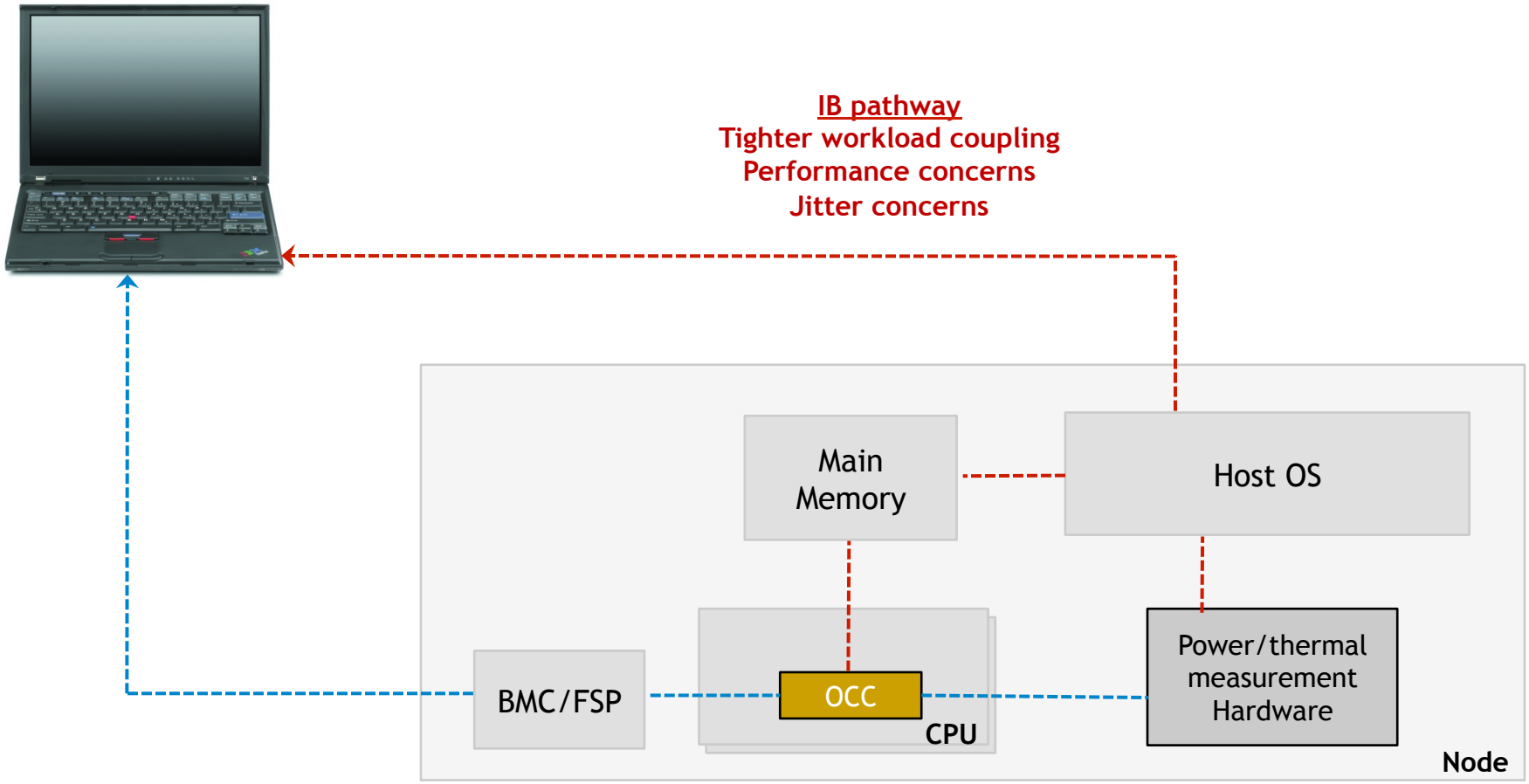
Used for longer Idle Periods
Power OFF the core plus private L2 cache.
Leave shared L3 cache running.
Saves ~ 80% power with < Y**m**s Latency

Winkle



Used for deconfigured cores
Power OFF the entire chiplet.
Requires restore/re-init to wakeup.
Shared L3 performance loss.
Saves ~ 95% power with < ZZ**m**s Latency

Measurement – In Band vs Out of Band



OOB pathway
Measure without affecting system performance
Difficult to correlate to system jobs/events

Measurement Options

FSP Based Systems – P7 and some P8 systems

Out of Band

- CIM – Buffers of 30 second readings. System Power/Thermal only
- IPMI/DCMI – System power/thermal only
- AMESTER – Detailed, Power/thermal/health/performance

In Band

- HMC – Performance Monitor

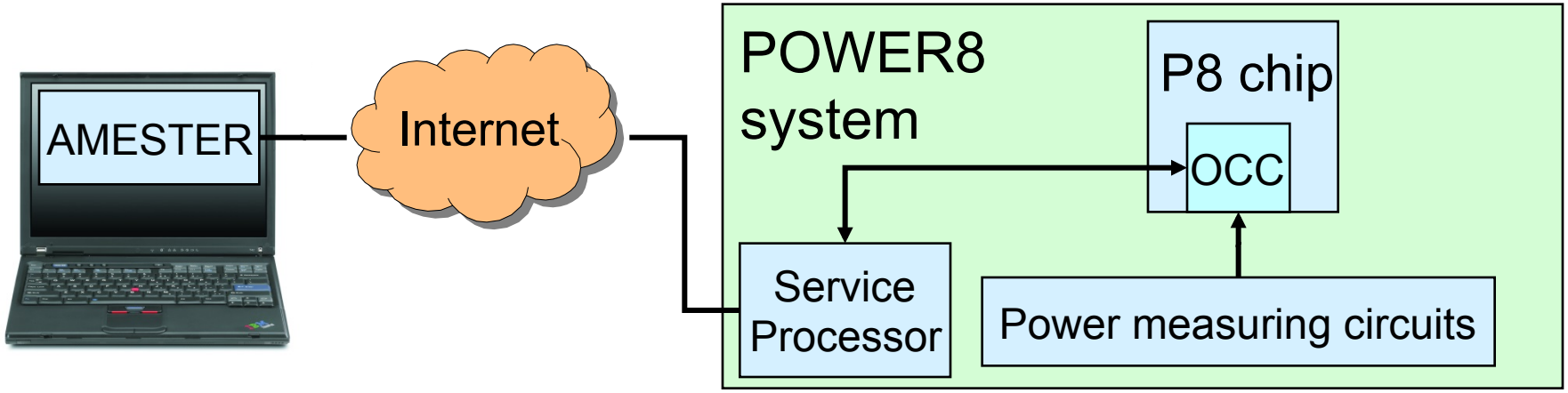
BMC Based Systems – Open Source P8

Out of Band

- IPMI/DCMI -- System, Every Power Rail, Core temperatures
- AMESTER – Open Source (coming soon)

AMESTER: Automated Measurement of Systems for Energy and Temperature Reporting – Out of Band

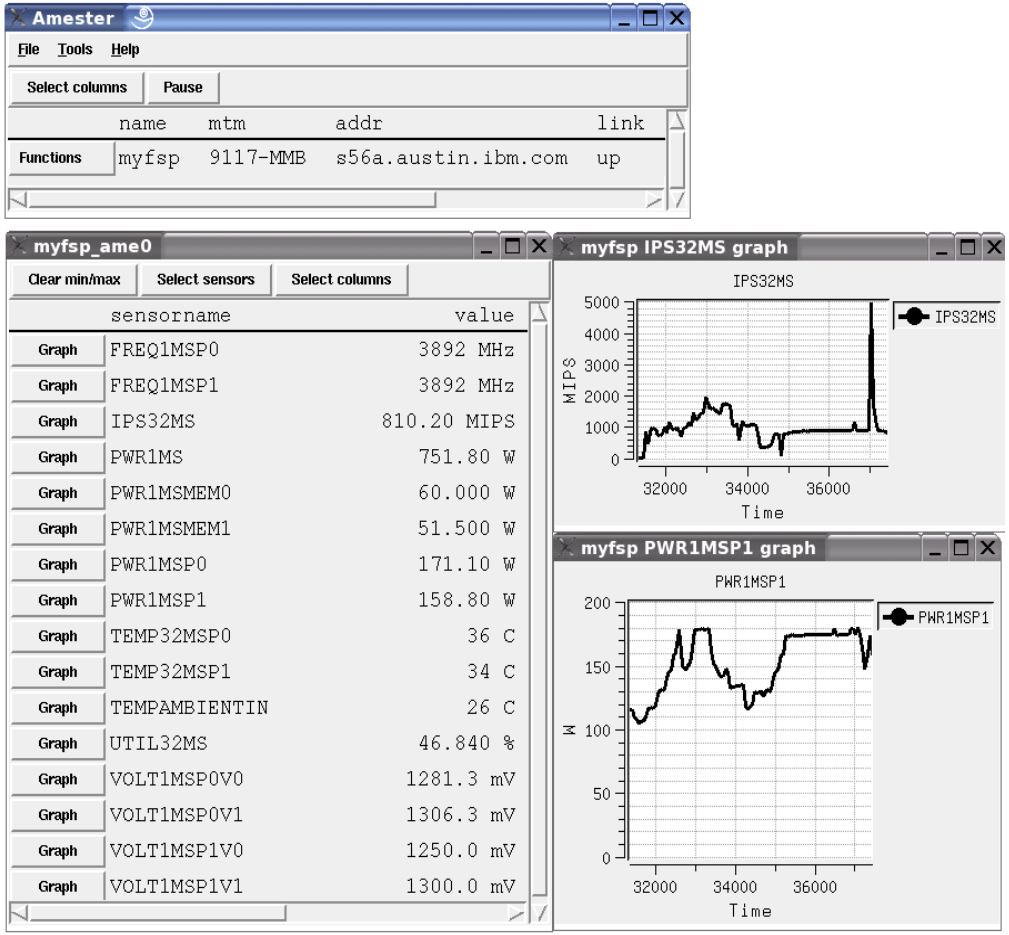
- Scriptable research tool that runs on a laptop or server (Windows/Linux).
- Connects to remote system to measure
- Uses management network and service processors
  - Does not interfere with operating system or workloads
  - No performance loss
- POWER8 On-Chip Controller (OCC).
  - POWERPC 405 processor
  - Firmware for power management
  - Implements AMESTER command set
  - No installation required on POWER



© 2011 IBM Corporation

AMESTER overview for POWER8

- Sensor data collection
  - Whole system power measurement
    Component power on POWER: Processor Vdd, Memory, IO, Storage, Fans
  - Core: temperature, frequency, utilization, memory bandwidth, instructions per second.
  - Histograms
- 250 µs tracing into 8KB buffer
  - E.g.1 sensor for 1 second
- 2 ms tracing into 8KB buffer
  - E.g. 1 sensor for 8 seconds
- Amester is a Tcl interpreter
  - All functions of Amester can be scripted by Tcl scripts
  - Data library to collect and graph user data
  - Output to CSV file for import into Excel or Matlab
- Parameter interface, provides ability to read/ write key firmware variables
  - Core frequency override



© 2011 IBM Corporation

Sensor Description

FIELD	SIZE (BYTES)	DESCRIPTION
name	16	Sensor name
units	4	Sensor units of measurement
gsid	2	Global sensor ID – assigned by its constructor
sample	2	Latest sample of this sensor
sample_min	2	Minimum value since last reset
sample_max	2	Maximum value since last reset
freq	4	Update frequency (for Amester use)
scale_factor	4	Scaling factor (for Amester use)
type	1	Sensor type (e.g., power, thermal, utilization, etc.)
location	1	Sensor location (e.g., system, processor, core, etc.)
number	1	Relative sensor number within the same type/location code
status	1	Status and control register
accumulator	4	Accumulator register for this sensor
src_accum_snapshot	4	Copy of the source sensor's accumulator – used for time-derived sensors
update_tag	4	Count of the number of 'ticks' that have passed between updates to this sensor – used for time-derived sensors
sample_max_diff	2	Maximum sample difference since last reset
vector	pointer	Pointer to vector control structure. NULL if this is not a vector sensor.
mini_sensor	pointer	Pointer to entry in mini-sensor table. NULL if this sensor does not have a mini-sensor.

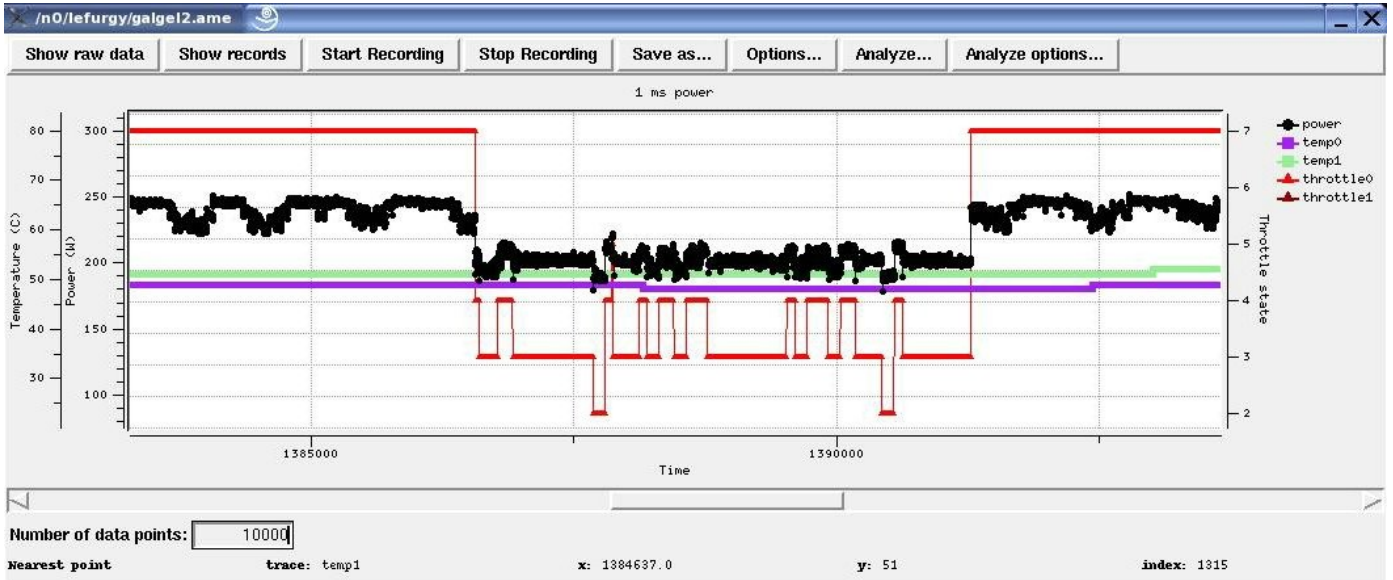
Sample Sensor list

440 Total Sensors – Power, Thermal, Performance, utilization, health

# / SENSOR	SENSOR NAME	RESOLUTION & UNITS	TIME SCALE	LOCATION	TYPE	DESCRIPTION
12	CUR250USVDDP0	0.01 Amps	every 250 usec	Processor	Sensor	Current consumption at Processor's Vdd Regulator Output
1	FANSPEEDAVG	1 RPMS	variable	System	Sensor	Average of all non-zero fan speeds - Master only sensor
12	FREQ250USP0Cy	1 MHz	every 250 usec	Core	Sensor	Requested frequency from OCC for Core y
12	FREQA250USP0	1 MHz	every 250 usec	Processor	Vector Sensor	Average of all core frequencies for Processor 0
12	FREQA250USP0Cy	1 MHz	every 250 usec	Core	Sensor	Average/actual frequency for Processor 0, Core y based on OCA data
1	IPS2MSP0	0.2 Mips	every 2 msec	Processor	Vector Sensor	Vector sensor that takes the average of all the cores in Processor x
8	IPS2MSP0Cy	0.2 Mips	every 2 msec	Core	Sensor	Instructions per second for Processor x, Core y
1	LOCDIMMAXP0Mx	integer (0-8)	every 2 msec	taur/on-chip equiv	Sensor	Location of the hottest DIMM temperaure behind Centaur x for Processor 0
1	M4RD2MSP0MxCyPz	0.01 Mrps	every 2 msec	taur/on-chip equiv	Sensor	Memory cached (L4) read requests per sec for Processor's MC x, Centaur y, Port-pair z (MBA
1	M4WR2MSP0MxCyPz	0.01 Mrps	every 2 msec	taur/on-chip equiv	Sensor	Memory cached (L4) write requests per sec for Processor's MC x, Centaur y, Port-pair z (MBA
1	MAC2MSP0MxCyPz	0.01 Mrps	every 2 msec	taur/on-chip equiv	Sensor	Memory activation requests per sec for Processor's Memory Controller x, Centaur y, Port-pair
1	MCPIFD2MSP0Cy	0.01 cpi	every 2 msec	Core	Sensor	Memory cycles per instruction frequency dependent for Processor x, Core y
1	MCPIFI2MSP0Cy	0.01 cpi	every 2 msec	Core	Sensor	Memory cycles per instruction frequency independent for Processor x, Core y
1	MEMSP2MS	0.10%	every 2 msec	taur/on-chip equiv	Vector Setting	Processor socket-level (summary) main memory throttle-level setting
1	MEMSP2MSP0	0.10%	every 2 msec	Processor	Vector Sensor	Vector sensor that takes the minimum of all the channel-pairs in this Processor
1	MEMSP2MSP0IGx	0.10%	every 2 msec	taur/on-chip equiv	Vector Setting	Memory interleave-group level main memory throttle setting, x=0..7 (?)
1	MEMSP2MSP0MxCyPz	0.10%	every 2 msec	taur/on-chip equiv	Setting	Centaur-level memory throttle setting (speed), y=0..7, z=0/1
1	MEMSP2MSP0MyCz	0.10%	every 2 msec	Channel-pair	Sensor	Final speed request for Processor's Memory Controller y, Channel-pair z
1	MIRB2MSP0MxCyPz	0.01 M events/s	every 2 msec	taur/on-chip equiv	Sensor	Memory Inter-request arrival idle intervals, Processor's MC x, Centaur y, Port-pair z (MBA01/M
1	MIRC2MSP0MxCy	0.01 K events/s	every 2 msec	taur/on-chip equiv	Sensor	Memory Inter-request arrival idle interval longer than programmed threshold at Processor's M
1	MIRH2MSP0MxCyPz	0.01 K events/s	every 2 msec	taur/on-chip equiv	Sensor	Memory Inter-request arrival idle intervals longer than high threshold, Processor's MC x, Cen
12	MIRL2MSP0MxCyPz	0.01 M events/s	every 2 msec	taur/on-chip equiv	Sensor	Memory Inter-request arrival idle intervals longer than low threshold, Processor 's MC x, Cent
1	MIRM2MSP0MxCyPz	0.01 K events/s	every 2 msec	taur/on-chip equiv	Sensor	Memory Inter-request arrival idle intervals longer than medium threshold, Processor's MC x,
1	MLP2P0MxCy	1 Events/s	every 2 msec	taur/on-chip equiv	Sensor	Number of LP2 exits at Processor's MC x, Centaur y
12	MPU2MSPwMxCyPz	0.01 Mrps	every 2 msec	taur/on-chip equiv	Sensor	Memory power-up requests per sec for Processor's Memory Controller x, Centaur y, Port-pair
1	MRD2MSP0Mx	0.01 Mrps	every 2 msec	Memory	Sensor	Memory read requests per sec for Processor's Memory Controller x
12	MSTL2MSP0Cy	0.01 cpi	every 2 msec	Core	Sensor	Memory (L1) stalled cycles per instruction for this Processor, Core y
1	MTS2MSP0MxCy	count	every 2 msec	taur/on-chip equiv	TimeStamp	Last received Timestamp (frame count) from Centaur
1	MWR2MSP0Mx	0.01 Mrps	every 2 msec	Memory	Sensor	Memory write requests per sec for Processor's Memory Controller x
1	PPICP0Cy	0.01%	every 250usec	Core	Sensor	Percentage of Prevented Instruction Cycles due to Throttling Events
1	PRCDupdatedur	1 usec	every 250 usec	Processor	Sensor	Time that is used to run through all the internal OCC procedures every 250usec
1	PWR250US	0.1 Watts	every 250 usec	System	Sensor	Bulk power of the system - Master only sensor
1	PWR250USFAN	0.1 Watts	every 250 usec	System	Sensor	Power consumption of the system fans - Master only sensor
1	PWR250USIO	0.1 Watts	every 250 usec	System	Sensor	Power consumption of the IO subsystem (including storage digital 5Volt or less rail) - Master
1	PWR250USMEM0	0.1 Watts	every 250 usec	Processor	Sensor	Power consumption for Memory for this Processor
12	PWR250USP0	0.1 Watts	every 250 usec	Processor	Sensor	Power consumption for this Processor
8	PWR250USSTORE	0.1 Watts	every 250 usec	System	Sensor	Power consumption of the storage subsystem (storage 12Volt rail) - Master only sensor
8	PWR250USVCSP0	0.1 Watts	every 250 usec	Processor	Sensor	Power consumption for this Processor's Vcs Regulator Input (12Volt)
16	PWR250USVDDP0	0.1 Watts	every 250 usec	Processor	Sensor	Power consumption for this Processor's Vdd Regulator Input (12Volt)
16	PWRPX250USP0Cy	0.1 Watts	every 250usec	Core	Sensor	Power Proxy sensor for Processor 0, Core y
8	SCOMreadduration	1 usec	every 250 usec	Processor	Sensor	Time duration for GUPI engine to complete reading all the SCOMs in a 250usec interval

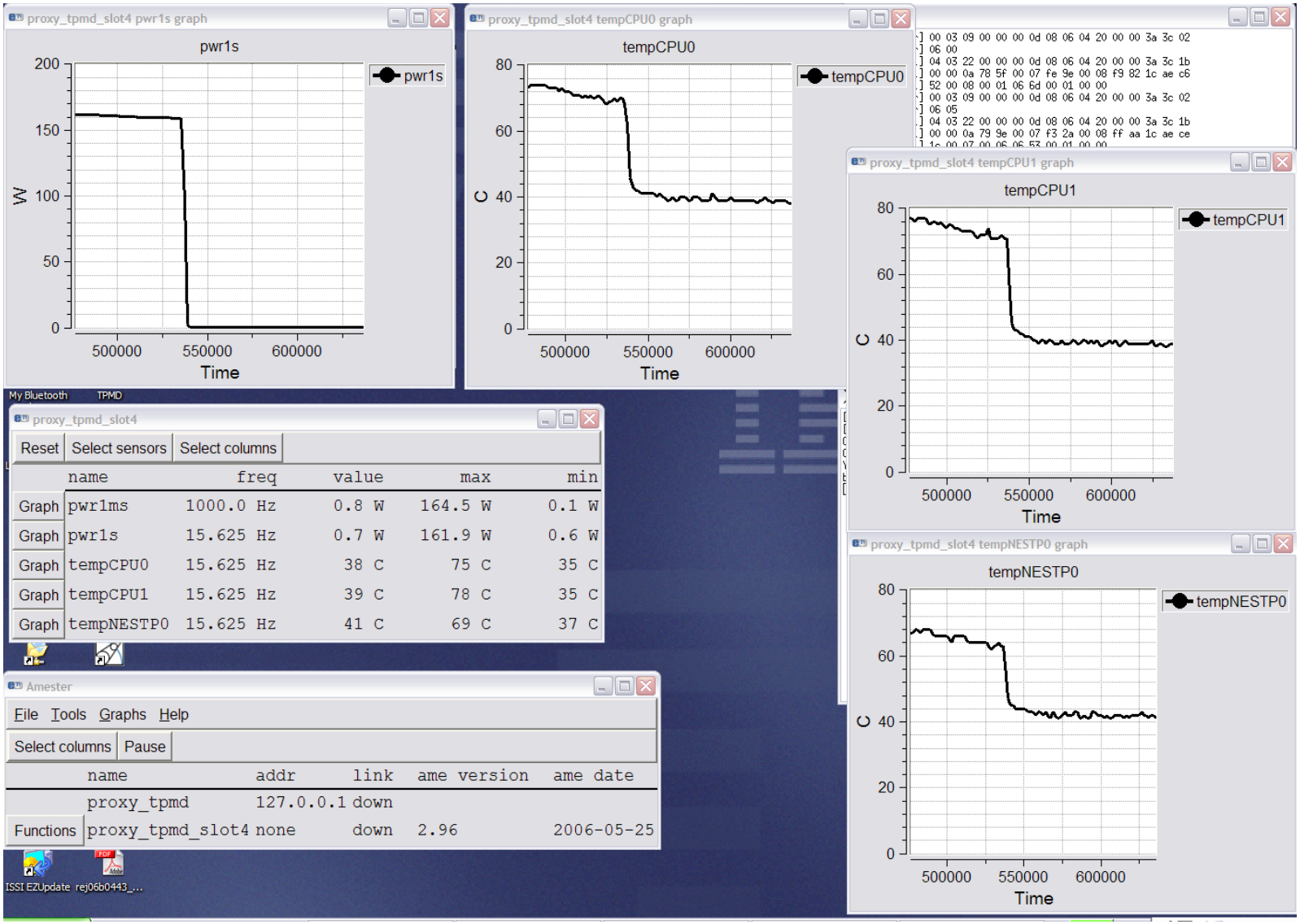
Insights

- Visualization is key to rapid prototyping and problem solving
- Correlation of power consumption with other metrics
- Time-alignment of sensor data is crucial for debugging firmware algorithms
- Examples
  - Measuring settling time of power capping controller after workload changes
  - Developing dynamic voltage/frequency scaling algorithms
  - Discovering small non-steady behavior when steady-state behavior was expected



© 2011 IBM Corporation

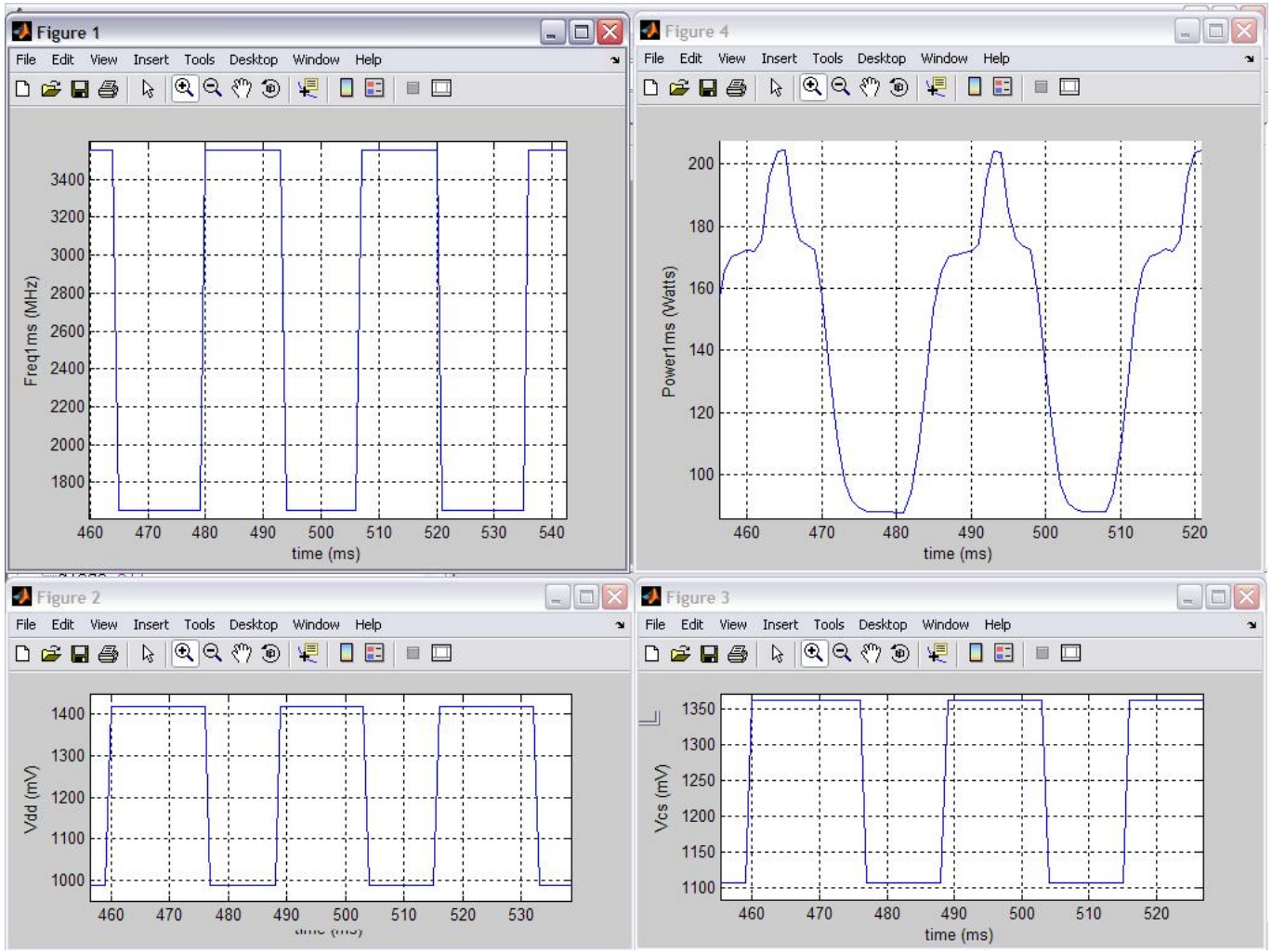
Collecting sensor data for a POWER6 system



© 2011 IBM Corporation

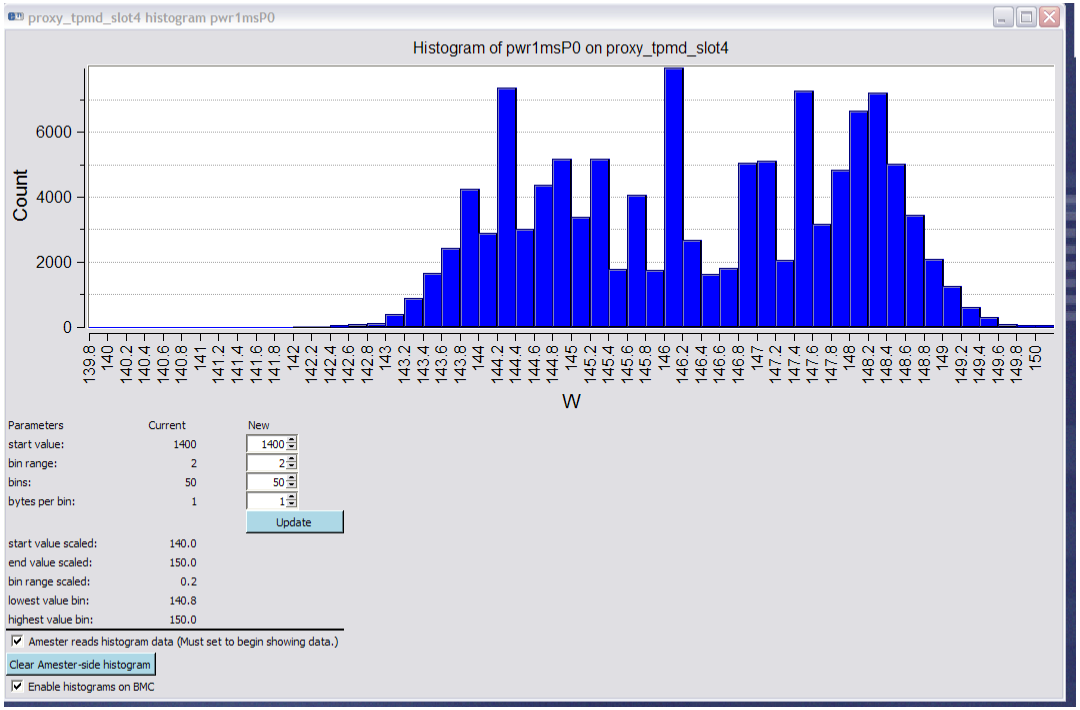
POWER7 voltage slew study

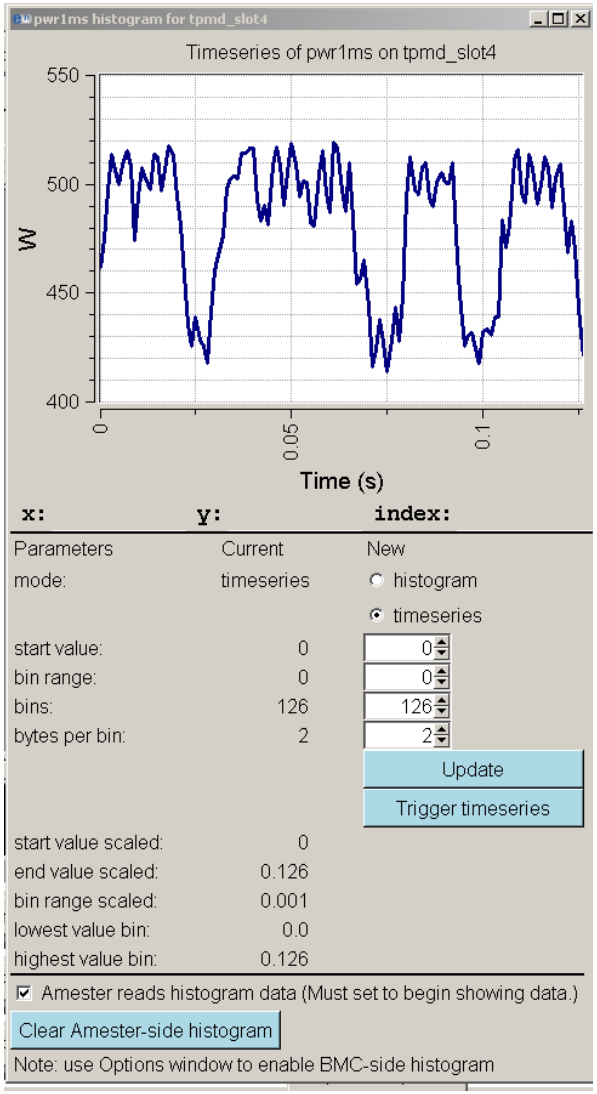
- Data imported to Matlab for graphing



© 2011 IBM Corporation

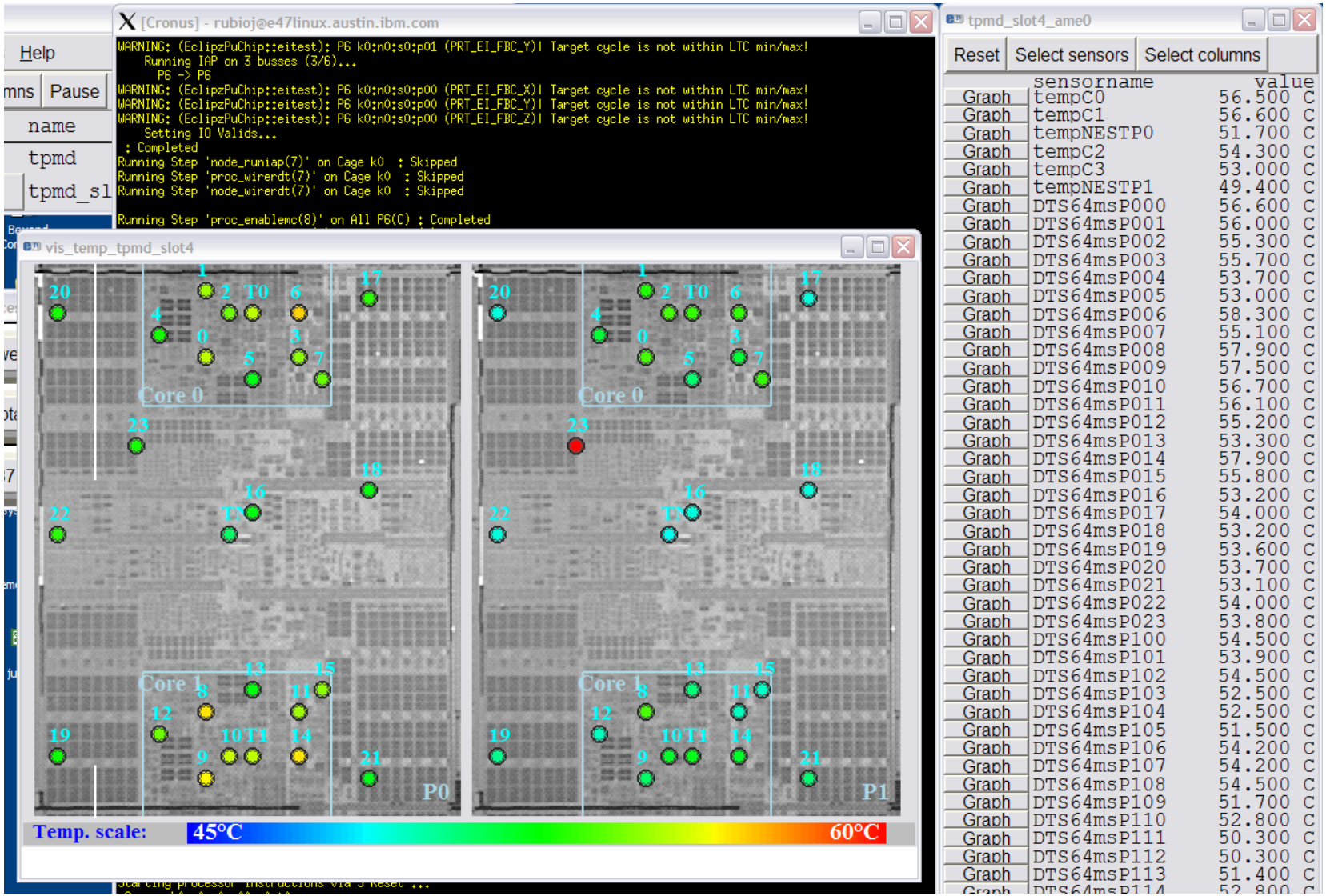
POWER7 histograms and traces of any sensor





© 2011 IBM Corporation

POWER6 core temperature measurement (custom script)



© 2011 IBM Corporation

What's Next

- GPU incorporation
- Open source AMESTER
- In band AMESTER
- Processor on/off – latency improvements
- Higher peak frequencies

Summary

- Power and Performance are tightly linked
  - Power consumption is a large portion of TCO
  - Performance is king
- Measurement at the right granularity is key
  - Measure, then control and optimize
- Visualization speeds analysis
- Open POWER enables innovation

TYAN'S OPENPOWER CUSTOMER REFERENCE SYSTEM

Innovative, Collaborative and Open

TYAN's OpenPOWER Customer Reference System - Innovative, Collaborative and Open

Open resources, management flexibility, and hardware customization are becoming more important to IT experts across various industries. To meet the emerging needs of evolving IT worlds, TYAN is honored to present its Palmetto System, the **TYAN GN70-BP010**. As the first commercialized customer reference system provided from an official member from the OpenPOWER ecosystem, the **TYAN GN70-BP010** is based POWER 8 Architecture and follows the OpenPOWER Foundation's design concept.

The **TYAN GN70-BP010** is a customer reference system which allows end users to deploy software based on the OpenPOWER architecture tailored to their individual requirements. It provides another opportunity for users to run their applications in a way of cost effective and flexible way. It is an innovative and collaborative hardware solution for IT experts who are looking for a more open, flexible, customized, and intelligent IT deployment.

GN70-BP010

USD$2850 FOB H.K.

Bundle

- (1) IBM® Power 8 Turismo SCM processor
- (1) Passive-Heatsink
- (4) 4GB, DDR3L-1600MHZ memory DIMMs
- (1) 500GB 3.5" HDD

References

Open Power Blog link:

- http://openpowerfoundation.org/press-releases/occ-firmware-code-is-now-open-source/

GitHub pages:

- OpenPOWER project: https://github.com/open-power
- OCC: https://github.com/open-power/occ
- Building OpenPOWER: https://github.com/open-power/op-build
- New AMESTER open source link

EnergyScale White Paper

http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=SA&subtype=WH&htmlfid=POW03125USEN

Youtube CIM link: https://www.youtube.com/watch?v=SsKE1Vg7PPg&feature=youtu.be

Programming Amester

- Amester is 40K lines of [incr Tcl]
  - Object-oriented Tcl
  - Uses Tk and BLT libraries for GUI
- Extend with Tcl, [incr Tcl], or C code
  - Can replace existing functions on-the-fly (debug as you run!)
  - Complete control
- Distributed as Tcl starpack
  - Everything is wrapped in a single binary image
    - Tcl interpreter binary, all Tcl libraries, Amester application, documentation
  - Simple to install and uninstall
- Amester commands on measured server finish in under 1ms
- Amester round-trip command latency is 50-300 ms, depending on the service processor firmware and network latency.

© 2011 IBM Corporation