MODELING PEP2.0

Michael Quaranta

IBM Systems Expert Labs

WHY RUN A PEP2.0 SIMULATION?

- When we talk about PEP2.0, or the *IBM Power Private Cloud with Shared Utility Capacity,* sometimes we tend to focus only on the "shared" aspect, i.e. the savings you get from pooling and sharing of Power systems' resources.
- This of course can be a significant savings in and of itself, especially where there is standby or DR capacity that is often idle.
- The pool savings can be compelling - but it is not the only savings a PEP2.0 customer can achieve.

WHY RUN A PEP2.0 SIMULATION?

- The IBM Power Private Cloud with Shared Utility Capacity brings cloud economics to a customer's enterprise.
- We tend to think of the comparison of "invested" resources vs. "pay-as-you-go" as an CapEx vs. OpEx discussion.
- While pay-as-you-go can certainly be a preference for some customers, the metered consumption (consumption in excess of invested "base" resources) doesn't have to be billed in arrears.
- With the capability to purchase Capacity Credits, customers can draw down from these consumption credits on account.
  - So these could certainly be bought in advance as a capital expenditure.

SO WHY USE METERED CONSUMPTION WHEN IT'S MORE EXPENSIVE PER MINUTE THAN INVESTED CONSUMPTION?

- The answer lies in finding out "how utilized is the Nth core?"
  - In other words, if I invest in a core, it has an effective per-minute cost. But what if I'm not using it every minute of every hour of every day? Maybe the first or second core in a pool of base activations is always in use, but what about the cores that are near the aggregate peak?
- If you knew a pay-as-you-go core costs twice as much as an invested core, would it be worth it if that core was used
  - 10% of the time?
  - 20% of the time?
- At a 2:1 ratio we would think that any core used less than half the time is better paid for as a metered core, rather than investing in a core that is under-utilized.

STATISTICS 101: MEAN ≠ MEDIAN

- A common misconception "half the class is below average"
- Median (or 50th Percentile) means half of the data points are above and half are below that median
- I once lived in a town of 50,000 people but one was a billionaire. Average net worth was way higher than median in this case.

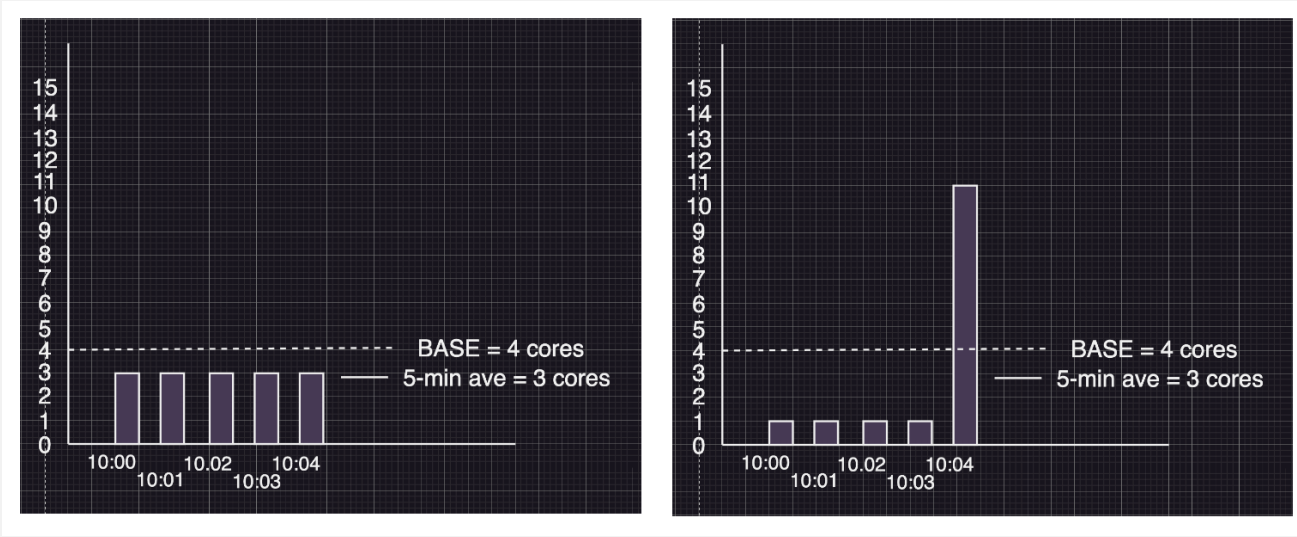
COMPARE

```
60 month cost per              base core = $ 7950.00 (activation)
                     + $   18.55 * 60 = $ 1113.00 (monthly hw per core maint)
                     + $ 1955.40 *  5 = $ 9777.00 (annual SW cost)
                                        ---------
                                        $18840.00
```

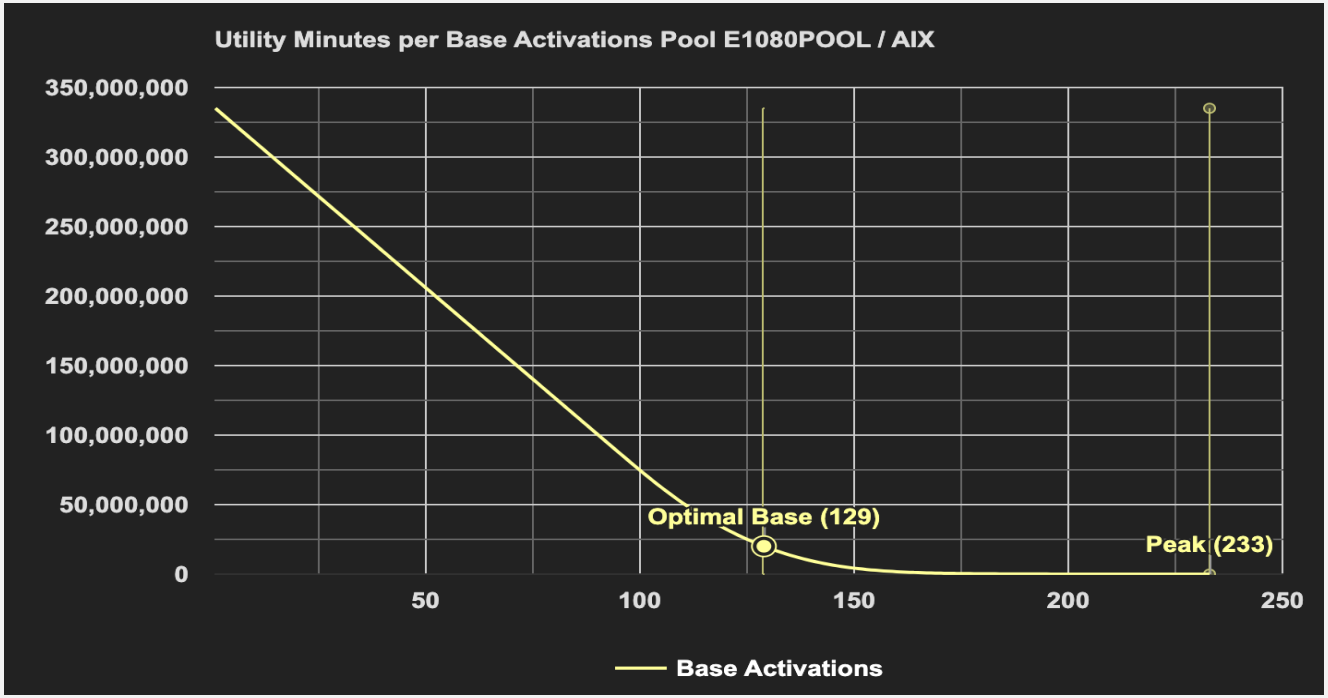
```
utility price per minute: $    0.0073 (activation cost)
                          $    0.0085 (system sw cost)
                          ------------
                          $    0.0157 (total)
```

Price per AIX core minute (invested) = $18,840/5/365/1440 = $0.0072
Price per AIX core minute (pay-as-you-go) = $0.0157
i.e. approximately twice as expensive to pay-as-you-go
THUS: any core needed less than half the time – better to pay-as-you-go

WHY CAN'T WE USE 5-MINUTE SAMPLES



TYPICAL UTILITY MINUTES PER BASE CHART



GOAL: FIND THE "SWEET SPOT"

- Adding more base above this "sweet spot" is wasteful, as base activations are underutilized
- Having less base than the sweet spot means you would pay a premium for the metered cores

GATHERING DATA FOR THE STUDY

- PEP2.0 is metered by the minute, so we want one-minute samples of PhysC:
  - physC = Amount of CPU consumed by an LPAR (whether busy or idle)

POWER HYPERVISOR: TIME BASE REGISTER

- The hypervisor gathers data from the Power Systems' Time Base (TB) register.
- The TB on the server ticks at the rate of 512,000 ticks per ms.
- A virtual processor (VP) has up to 8 SMT threads.
- When a VP is dispatched, PHYP record the current TB value.
- When all of the 8 SMT threads have nothing to do or the VP is preempted, the Power hypervisor records the ending TB.
- So, each dispatch accumulates TB cycles.
  - There is a TB register for every thread on the system and all have exactly the same value at any moment in time. The HMC's lslparutil command retrieves these counts is based on the TB measured dispatch times. Using this method of capturing lslparutil counters allows an agent-less, OS-agnostic collection of the values needed to calculate physC.

USING THE COUNTERS

- physC = (Δ capped_cycles + Δ uncapped_cycles) / Δ time_cycles
  - **capped_cycles** = The number of capped processing cycles utilized by this partition since the managed system was started.
  - **uncapped_cycles** = The number of uncapped processing cycles utilized by this partition since the managed system was started.
  - **time_cycles** = The number of time cycles since the managed system was started.

HMC AND THE COUNTERS

- The HMC keeps these counters in a datastore, but it is quite volatile.
- The HMC is not meant to warehouse this data, so it prunes it after a few days.
  - For that reason, the data is best offloaded.
  - Monitoring tools like ITM, LPAR2RRD and the Cloud Management Console can store these samples for much longer retention periods.
- If we can find a tool that has one-minute physC samples, we can harvest data to use for the study.