Now that we know how to specify the focus metric

- We will look at
  - The [go] statement `measure=on` option and its subparameters
    - Specifying `measure=on` on a Go statement means "watch the focus metric and when you have seen enough to make a measurement of the specified accuracy, stop. Timeout if it takes too long."
  - The [go] statement `dfc=pid` option and its subparameters
    - If you don't specify a `dfc`, the workload settings remain constant through the test.
    - If you specify a `dfc` (dynamic feedback controller), it gets called at the end of every subinterval once all the rollups are done.
    - The DFC looks at what has happened so far, looking at all workload data and all subsystem data, and then may use the ivy engine real time edit rollup mechanism to send out parameter updates to rollup instances (to the workload threads belonging to the rollup instance).

`measure = on`

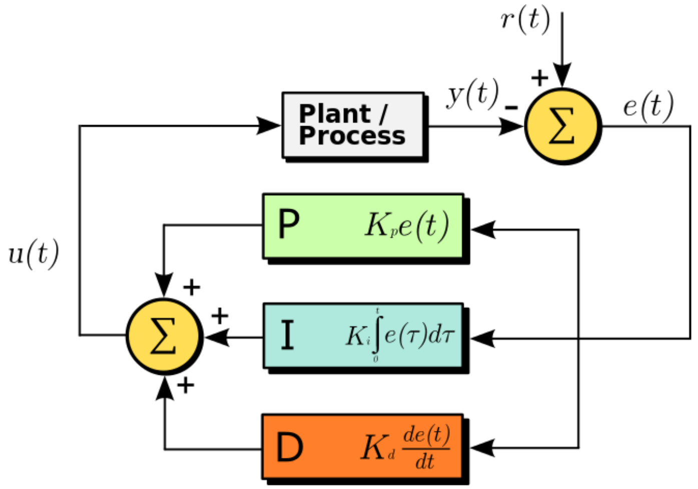
- `accuracy_plus_minus = "2%"`
  - Any numeric value with an optional trailing % sign maybe specified.

- `confidence = "95%"`
  - How confident you need to be that your measurement falls within the specified plus or minus range around the long term average that you would get measuring forever.
  - Default is 95%
  - Ivy has a menu of 11 specific pre-loaded confidence values that you pick from.
    - 50%, 60%, 70%, 80%, 90%, 95%, 98%, 99%, 99.5%, 99.8%, and 99.9%
    - http://en.wikipedia.org/wiki/Student%27s_t-distribution

`measure` Write Pending-based stability criteria

- `max_wp = "2%"` - default “100%”
  - A subinterval sequence will be rejected if WP is above the limit at any point in the sequence.
  - Set this to "1%' or so for read tests to ensure WP is empty during the test.
- `min_wp = "67%"` - default “0%”
  - A subinterval sequence will be rejected if WP is below the limit at any point in the sequence.
  - Use this for write tests to ensure WP is full during the test.
- `max_wp_change = "3%"` - default “3%”
  - A subinterval sequence will be rejected if WP varies up and down by more than the specified (absolute) amount at any point in the sequence. `max_wp_range="3%"` matches from 0% to 3% Write Pending, as well as from 67% to 70% Write Pending. (not a percent OF the WP value)
  - Use this in general all the time so you reject periods with major movement in Write Pending.

`dfc=pid` dynamically adjusts total IOPS



- General purpose DFC – see http://en.wikipedia.org/wiki/PID_controller
- The feedback is the value of the focus metric
  1. `source=workload`
     - E.g. host-view service time, response time
  2. `source=RAID_subsystem`
     - E.g. `subsystem_element="PG"`, `subsystem_metric="busy_percent"`.
- User specifies "`P`", "`I`", and "`D`" constants.

PID loop basics

- See "PID loop" on wikipedia   https://en.wikipedia.org/wiki/PID_controller
- ivy's PID loop dynamically adjusts IOPS up and down to hit a target value for the focus metric.
- The "error signal" is the difference between the measured focus metric value and the target value.

PID loop – computing new IOPS setting

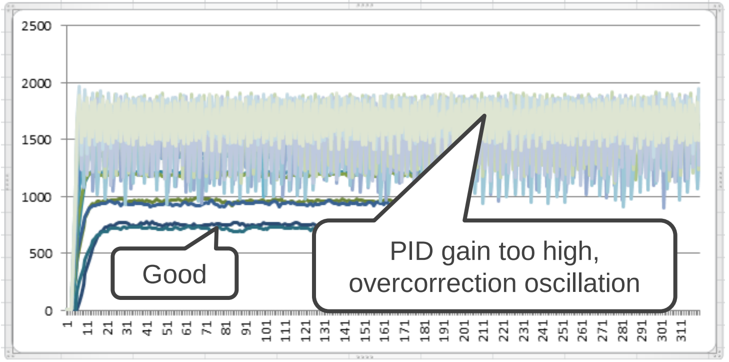
HITACHI
Inspire the Next

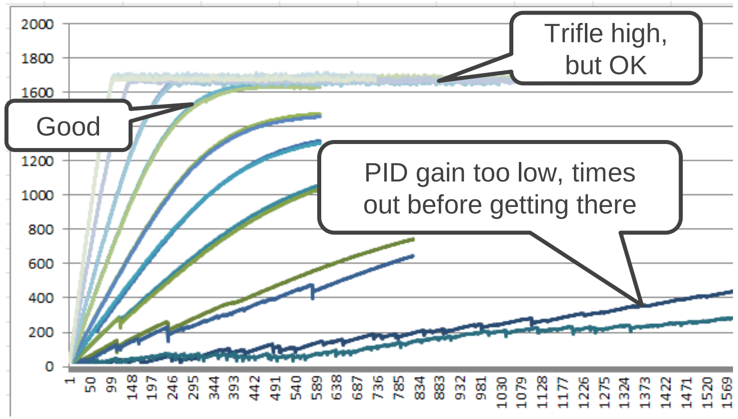
- The user provides 3 multiplier factor constants: p, i, d.
- The new total_IOPS setting is

– "`P`" times the error signal	(**p**roportional factor)
+ "`I`" times the sum of the error signal since the start of the test	(**i**ntegral factor)
+ "`D`" time the rate of change of the error signal	(**d**erivative factor)

- The ivy engine "edit rollup" mechanism sends out the new total_IOPS setting to the focus metric's rollup instance (usually "`all=all`"), where it takes effect in real time.

Tuning PID loop





- The PID loop mechanism works fine, but there's no guidance on how to program the `P, I, D` constants.
- Brief experimentation revealed that to plot a 20% busy something requires using different PID constants from plotting an 80% busy something.
- For `busy_percent` or `service_time` "`I`" constant alone gives good results.
- Anticipate that for `WP_percent`, would need to add use of "`D`" constant to gate speed of approach to WP target.
- Constants work over a broad range, but tuning constant values optimizes speed of "locking on" but with reasonable stability.
- Future idea: automatic gain control of PID constants

[Go] parameter summary

- Overall
  - stepname = stepNNNN
  - Subinterval_seconds = 5
  - warmup_seconds = 5
  - measure_seconds = 60
  - cooldown_by_wp = on
- For dfc = pid
  - p = 0
  - i = 0
  - d = 0
  - target_value = 0
- For measure = on
  - accuracy_plus_minus = "2%"
  - confidence = "95%"
    - 50%, 60%, 70%, 80%, 90%, 95%, 98%, 99%, 99.5%, 99.8%, or 99.9%
  - max_wp = "100%"
  - min_wp = "0%"
  - max_wp_change = "3%"
- Focus metric
  - focus_rollup = all
  - source = ""
    - or workload / RAID_subsystem
  - subsystem_element = ""
  - element_metric = ""
  - category = overall
    - or read, write, random, sequential, random_read, random_write, sequential_read, sequential_write
  - accumulator_type = ""
    - or bytes_transferred, service_time, response_time
  - accessor = ""
    - avg, count, min, max, sum, variance, standardDeviation