Correlator Redis Keys

Jack Hickish

June 2019

1 Introduction

The HERA correlator implements a correlation pipeline using the HASHPIPE engine. The HASHPIPE library has built-in support for status monitoring using a redis database. This document outlines the status keys the correlator posts to redis, most of which come from the original PAPER GPU correlator design.

2 X-Engine keys

X-engines store status hashes in redis using the keys: `hashpipe://<host>/<M>/status`, where 'host' is the X-engine hostname (px{1..16}) and M is the pipeline instance (0..1) running on the machine.

Each status key may be queried using REDIS's `HGET` or `HGETALL` commands. Hash entries are as described in the following table. All entries are stored as strings.

2.1 General Hash Entries

Key Name	Description
GIT_VER	The git hash of the running version of `paper_gpu`.
INSTANCE	The instance ID of this hashpipe pipeline.
TIMEIDX	The index (0, 1) of the time slices being processed by this pipeline. 0 = even, 1 = odd
XID	The overall ID of this X-engine pipeline in the total system (0..31)

2.2 Net-thread Hash Entries

Key Name	Description
BINDHOST	The Ethernet interface the network thread is bound to. Eg. "eth3"
BINDPORT	The UDP port the network thread is bound to
MISSEDFE	The number of antennas not being received over the network
MISSEDPK	The number of packets missed in the last data block, after taking into account the number of missing antennas
NETBKOUT	The block ID of the last completed data block
NETDROPS	Number of dropped packets reported by the kernel in the last data block
NETDRPTL	A running total of the number of dropped packets reported by the kernel
NETGBPS	The throughput (data + application header bits) achieved for the last processed data block, based on the time spend processing and receiving (but not waiting)
NETMCNT	The MCNT of the current block being processed
NETPKTS	? Number of packets received in the last data block
NETPKTTL	? Running total of the number of packets received
NETPRCMN	The minimum reported time (in ns) required to process a packet
NETPRCMX	The maximum reported time (in ns) required to process a packet
NETPRCNS	The average time (in ns) required to process a packet based on the last data block processed
NETRECMN	The minimum time (in ns) taken by the "get packet" call to return
NETRECMX	The maximum time (in ns) taken by the "get packet" call to return
NETRECNS	The average time (in ns) taken by the "get packet" call to return, based on the last data block processed
NETWATMN	The minimum time (in ns) spent waiting for a packet
NETWATMX	The maximum time (in ns) spent waiting for a packet
NETWATNS	The average time (in ns) spent waiting for a packet based on the last data block processed
NETSTAT	A simple descriptor of the thread status: "running", "blocked out", "holding"

2.3 Fluff-thread Hash Entries

Key Name	Description
FLUFBKIN	The current hashpipe block ID being processed
FLUFGBPS	The throughput (in gigabits per second of input processed) of the last processed hashpipe block
FLUFMCNT	The MCNT of the current block being processed
FLUFMING	The minimum reported throughput (in gigabits per second of input processed) of all the blocks processed so far
FLUFSTAT	A simple descriptor of the thread status: "fluffing", "blocked in", "blocked out". TODO: check that these are actually the values

2.4 GPU-thread Hash Entries

Key Name	Description
GPUBLKIN	The current hashpipe block ID being processed
GPUDEV	The ID of the GPU device being used by this thread
GPUDUMPS	A running total of the number of data dumps from the GPU
GPUGBPS	The throughput (in gigabits per second of **8 bit** input processed) of the last processed GPU block.
GPUMCNT	The MCNT of the current block being processed
GPUSTAT	A simple descriptor of the thread status: "waiting", "processing", "blocked in", "blocked out"
INTCOUNT	The number of spectra being accumulated on the GPU
INTSYNC	The MCNT to which integrations are synchronized
INTSTAT	Used to turn on and off GPU processing. "on", "off".

2.5 Output-thread Hash Entries

Key Name	Description
OUTBLKIN	Current input block ID being processed
OUTBYTES	Number of bytes transmitted for last integration
OUTDUMPS	Running total of number of integrations transmitted
OUTMBPS	Output data rate (in megabits per second) of last integration transmission
OUTSECS	Time taken to send last integration
OUTSTAT	Simple description of thread status "Processing", "blocked in"

3 Catcher Keys

The correlator catcher(s) store status hashes in redis using the keys: `hashpipe://<host>/<M>/status`, where 'host' is the catcher hostname (hera-sn{1..2}) and M is the pipeline instance (currently always 0) running on the machine.

Each status key may be queried using REDIS's `HGET` or `HGETALL` commands. Hash entries are as described in the following table. All entries are stored as strings.

3.1 General Hash Entries

Key Name	Description
General	
GIT_VER	The git hash of the running version of `paper_gpu`.
INSTANCE	The instance ID of this hashpipe pipeline.
SYNCTIME	The Unix time corresponding to MCNT = 0

3.2 Net-thread Hash Entries

Key Name	Description
Net thread	
BINDHOST	The Ethernet interface the network thread is bound to. Eg. "eth3"
BINDPORT	The UDP port the network thread is bound to
MISSEDPK	The number of packets missed in the last data block, after taking into account the number of missing antennas
NETBKOUT	The block ID of the last completed data block
NETDROPS	Number of dropped packets reported by the kernel in the last data block
NETDRPTL	A running total of the number of dropped packets reported by the kernel
NETGBPS	The throughput (data + application header bits) achieved for the last processed data block, based on the time spend processing and receiving (but not waiting)
NETMCNT	The MCNT of the current block being processed
NETPKTS	? Number of packets received in the last data block
NETPKTTL	? Running total of the number of packets received
NETPRCMN	The minimum reported time (in ns) required to process a packet
NETPRCMX	The maximum reported time (in ns) required to process a packet
NETPRCNS	The average time (in ns) required to process a packet based on the last data block processed
NETRECMN	The minimum time (in ns) taken by the "get packet" call to return
NETRECMX	The maximum time (in ns) taken by the "get packet" call to return
NETRECNS	The average time (in ns) taken by the "get packet" call to return, based on the last data block processed
NETWATMN	The minimum time (in ns) spent waiting for a packet
NETWATMX	The maximum time (in ns) spent waiting for a packet
NETWATNS	The average time (in ns) spent waiting for a packet based on the last data block processed
CNETSTAT	A simple descriptor of the thread status: "running", "blocked out", "holding"

3.3 Disk-thread Hash Entries

Key Name	Description
Disk Thread	
DISKBKIN	The current hashpipe block ID being processed
DISKGBPS	The throughput (in gigabits per second of input processed) of the last processed hashpipe block
DISKMCNT	The MCNT of the current block being processed
DISKMING	The minimum reported throughput (in gigabits per second of input processed) of all the blocks processed so far
DISKSTAT	A simple descriptor of the thread status: "idle", "blocked in", "blocked out". TODO: check that these are actually the values
DISKTBNS	? Average time (in ns) taken to transpose data, based on last block processed
DISKTMAX	? Maximum time (in ns) taken to transpose a data block
DISKTMIN	? Minimum time (in ns) taken to transpose a data block
DISKWBNS	? Average time (in ns) taken to write a block to disk
DISKWMAX	? Maximum time (in ns) taken to write a block to disk
DISKWMIN	? Minimum time (in ns) taken to write a block to disk
DUMPMS	? Time (in ms) required to ?? TODO: Suspect some of the above write time definitions are wrong
HDF5TPLT	Path to the HDF5 header template file
INTTIME	Number of spectra in each integration from the correlator
MSPERFILE	Target duration per output file (in ms)
NFILES	Number of files to be written in this recording run
NDONEFIL	Number of files written so far in this recording run
TAG	Ascii tag to be written into data files for this recording run
TRIGGER	Control signal to trigger a new data collection run