

Load Shedding for Network Monitoring Systems (SHENESYS)




Centre de Comunicacions Avançades
de Banda Ampla (CCABA)

Universitat Politècnica
de Catalunya (UPC)

UPC team: Pere Barlet-Ros
Josep Solé-Pareta
Josep Sanjuà
Diego Amores
Intel sponsor: Gianluca Iannaccone

Barcelona, February 3rd 2006

Agenda

- The scenario, challenges and objectives of SHENESYS
 - Work done and current status
 - Preliminary results
 - Work plan
 - Short term work plan
 - Other tasks
 - Equipment
 - Appendixes
 - Josep Sanjuàs: Intel performance counters
 - Diego Amores: Summary of his internship at Intel Research
- 

The scenario

- New network monitoring systems call for novel methods for
 - Expressing arbitrary queries
 - **Scheduling multiple competing queries**
- SHENESYS addresses the latter aspect
 - Schedule arbitrary queries in a resource constrained environment
 - Guarantee some level of quality of service
- Traditional resource management techniques are not viable
 - Push-based systems
 - Input data rates decided by external sources that cannot be controlled
 - Continuous input stream with extremely high data rates
 - Real-time constraints not only on responses but also in the input
 - Arbitrary computation
 - Incoming traffic unknown and unpredictable

Challenges

- Traffic is unpredictable and bursty in nature
 - Bursts can be several orders of magnitude higher than typical traffic
 - Provisioning to line speed might imply waste of resources
 - Bursts often produce different data than ordinary traffic
- Queries are unknown a-priori, arbitrary and complex
 - Resource over-provisioning is not a solution
 - Relational languages are usually not flexible enough to express even the simplest network queries
- Runtime profiling of resource usage is needed
 - Given a query resource consumption cannot be known before actually running it, even when knowing the input traffic
 - Need to understand correlation between traffic features and resource consumption of queries to be able to estimate resource consumption

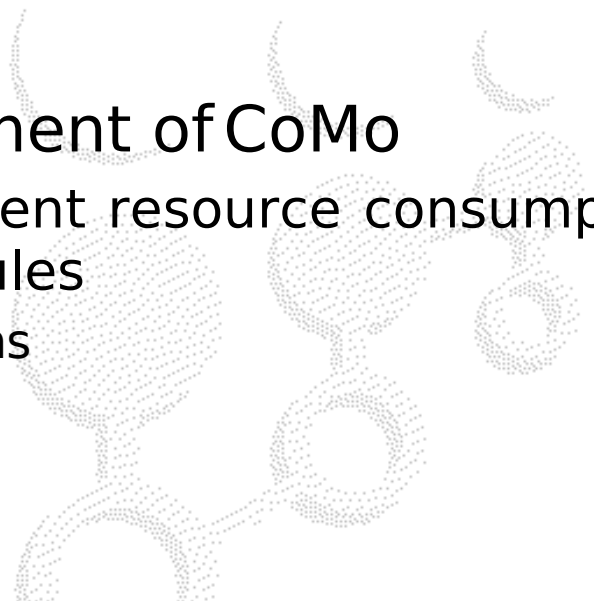
Challenges

- Provide QoS to arbitrary queries in a resource constrained environment
 - Network queries usually have QoS requirements in terms of response delay and accuracy
 - Not meeting QoS requirements can lead to useless results
- Robustness in front of network anomalies and attacks
 - Anomalies usually produce more resource consumption than usual
 - Monitoring systems are especially needed when network is at risk
 - Malicious users may try also to attack directly the monitoring system to cover up their actions


Objectives

- Predict resource usage of arbitrary queries
 - Profile CPU, memory and I/O usage of arbitrary queries
 - Find traffic features from packet stream that exhibit correlation with resource usage of queries
- Implement mechanisms to shed excess of load
 - Postpone or deny queries
 - Reduce accuracy of queries (e.g. via packet sampling)
 - Reuse or share computations among different queries
- Design and evaluate scheduling algorithms
 - Apply load shedding mechanisms to meet QoS requirements of *most* queries while maximizing the *utility* of the system
 - Utility can be a function of delay, accuracy and priority
 - Fast to early detect shortage of resources and avoid packet loss

Objectives

- Build a complex resource management system
 - Build a prototype in CoMo as a case study
 - Test robustness of resource management techniques in front of network anomalies and attacks
 - Contribute to the main development of CoMo
 - Build complex modules with different resource consumption patterns than existing CoMo modules
 - Identification of network applications
 - Anomaly and intrusion detection
 - Network forensic applications
 - Others
- 

Agenda

- The scenario, challenges and objectives of SHENESYS
 - **Work done and current status**
 - Preliminary results
 - Work plan
 - Short term work plan
 - Other tasks
 - Equipment
 - Appendixes
 - Josep Sanjuàs: Intel performance counters
 - Diego Amores: Summary of his internship at Intel Research
- 

Work done

- *Capture* operates on time bins
 - Ease the process of checking if there are enough resources to process a batch before the arrival of the next batch
 - Circular buffers were needed
 - Rewriting of libpcap and ERF sniffers
- On-line computing and logging of batch features
 - #pkts, #bytes, #unique_hashes_batch, #unique_hashes_table, #flushes_batch_will_cause, etc.
 - Probably more features will be needed for more complex modules
- On-line profiling and logging of CPU usage per module
 - TSC, system/userland cycles, L1, L2 and L3 (Xeon) cache misses, context switches, etc.
 - Callbacks (depend on the module)
 - Overhead (does not depend directly on the module)
 - Allocating memory, creating/flushing tables, etc.

Work done

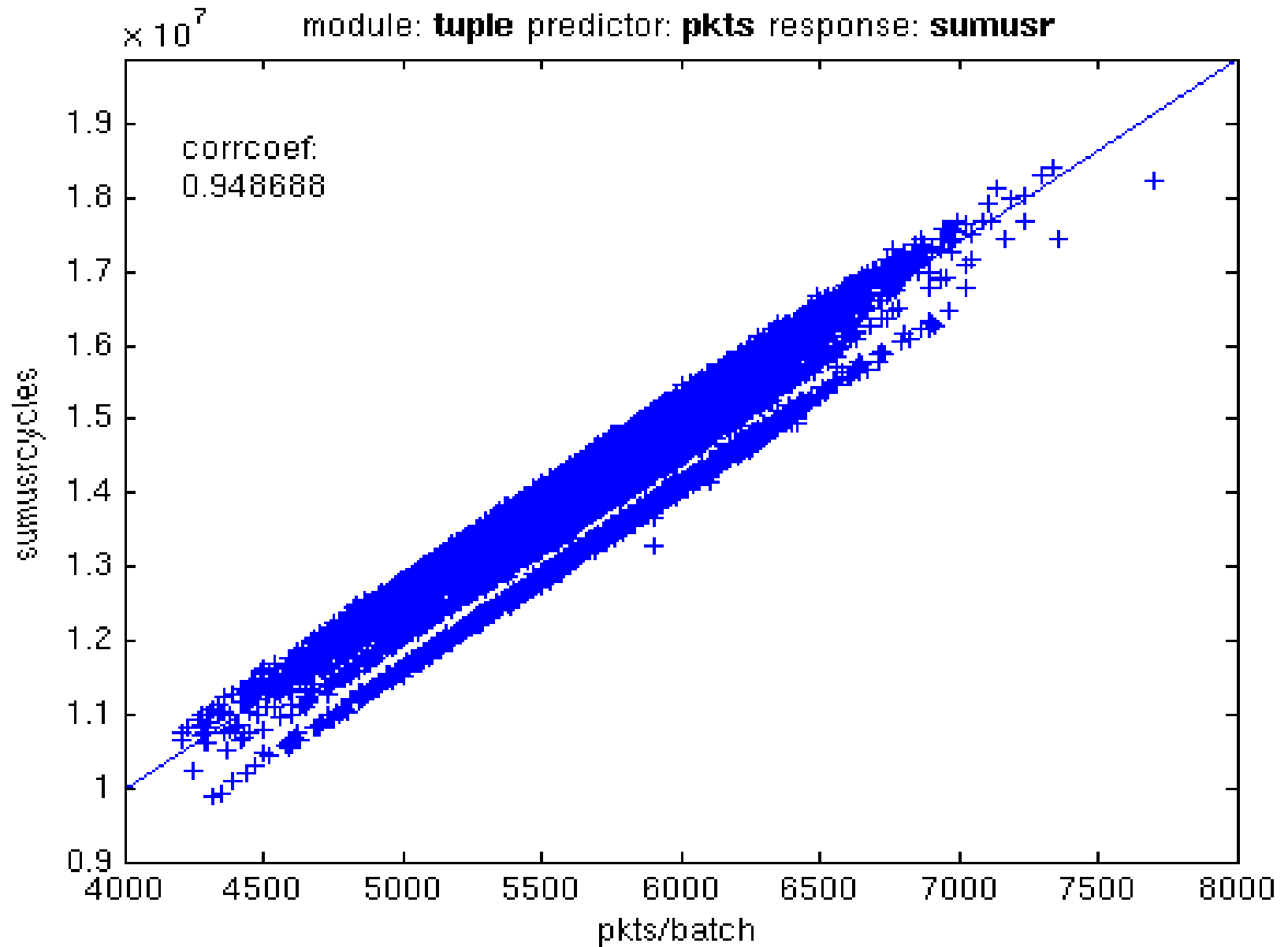
- Analysis of correlation between features of batches and CPU usage for standard modules
 - Tuple: #pkts, #unique_hashes_table
 - The rest: #pkts
 - #bytes is expected to matter for modules processing payloads (when collecting full packets)
- Analysis of techniques to predict CPU usage and study of prediction error
 - Prediction methods: Linear prediction, multiple linear prediction, etc.
 - All history, last 1 sec, 10 sec, 1 min, etc.

Agenda

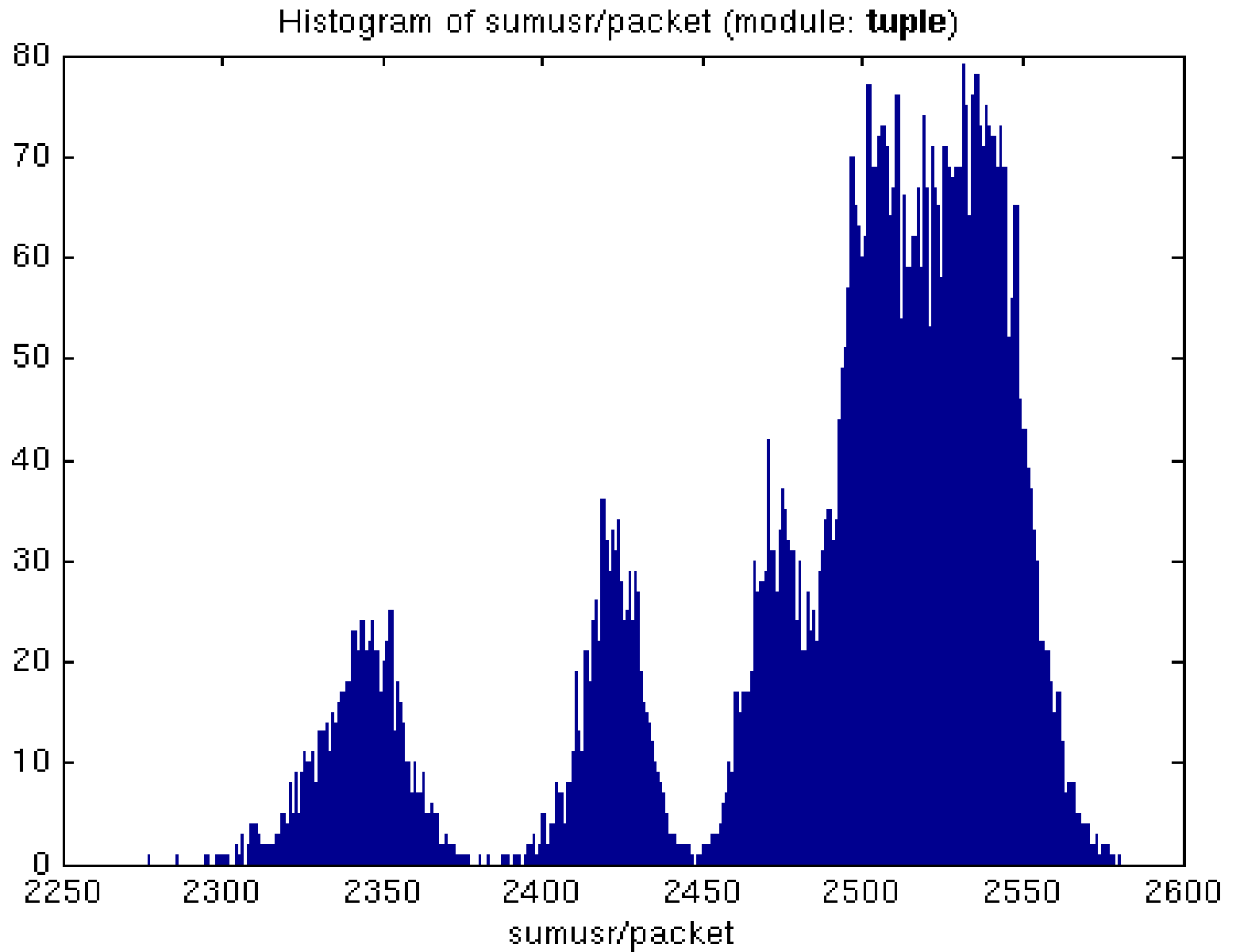
- The scenario, challenges and objectives of SHENESYS
- Work done and current status
- **Preliminary results**
- Work plan
- Short term work plan
- Other tasks
- Equipment
- Appendixes
 - Josep Sanjuà: Intel performance counters
 - Diego Amores: Summary of his internship at Intel Research



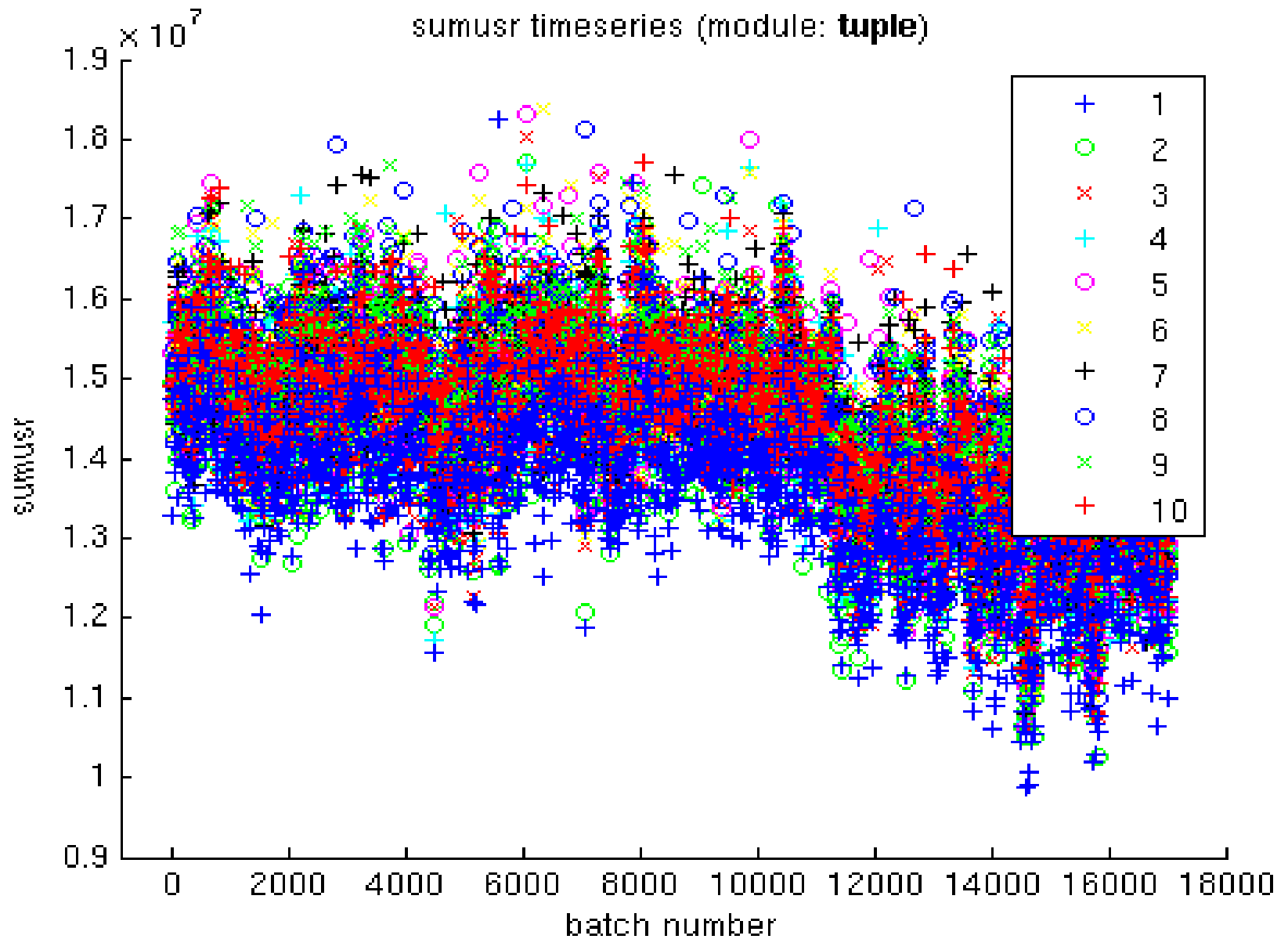
Callback cycles



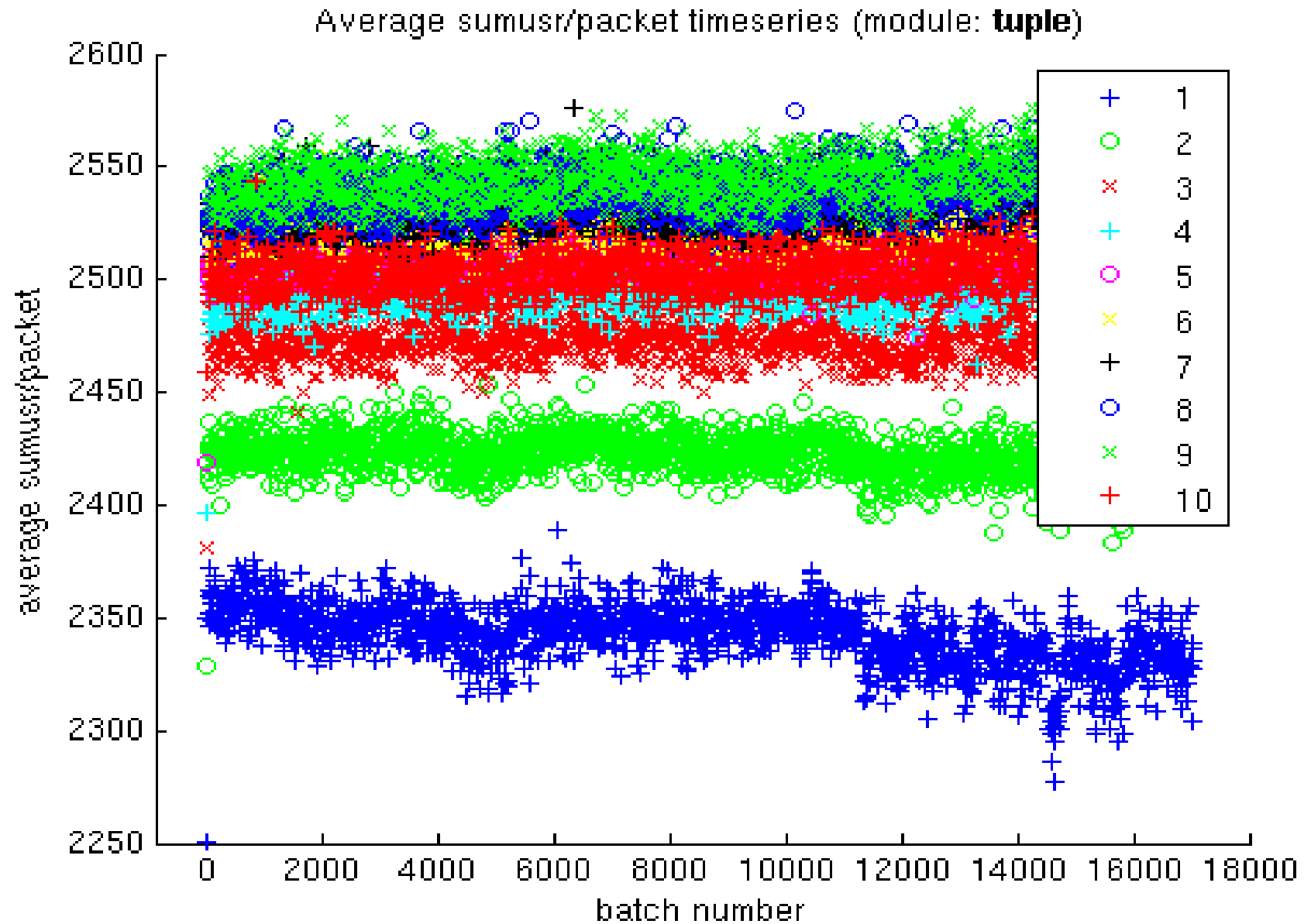
Callback cycles



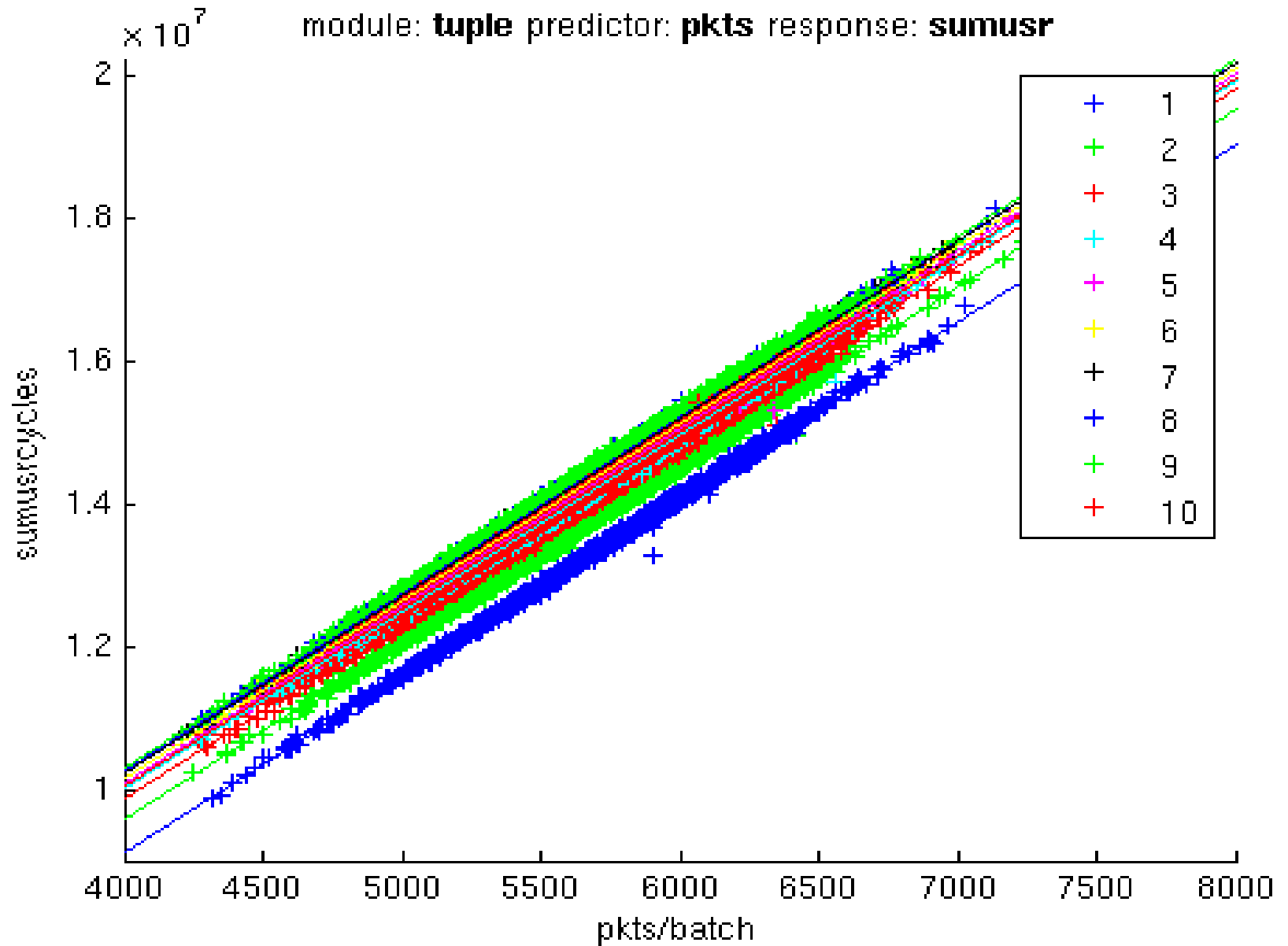
Callback cycles



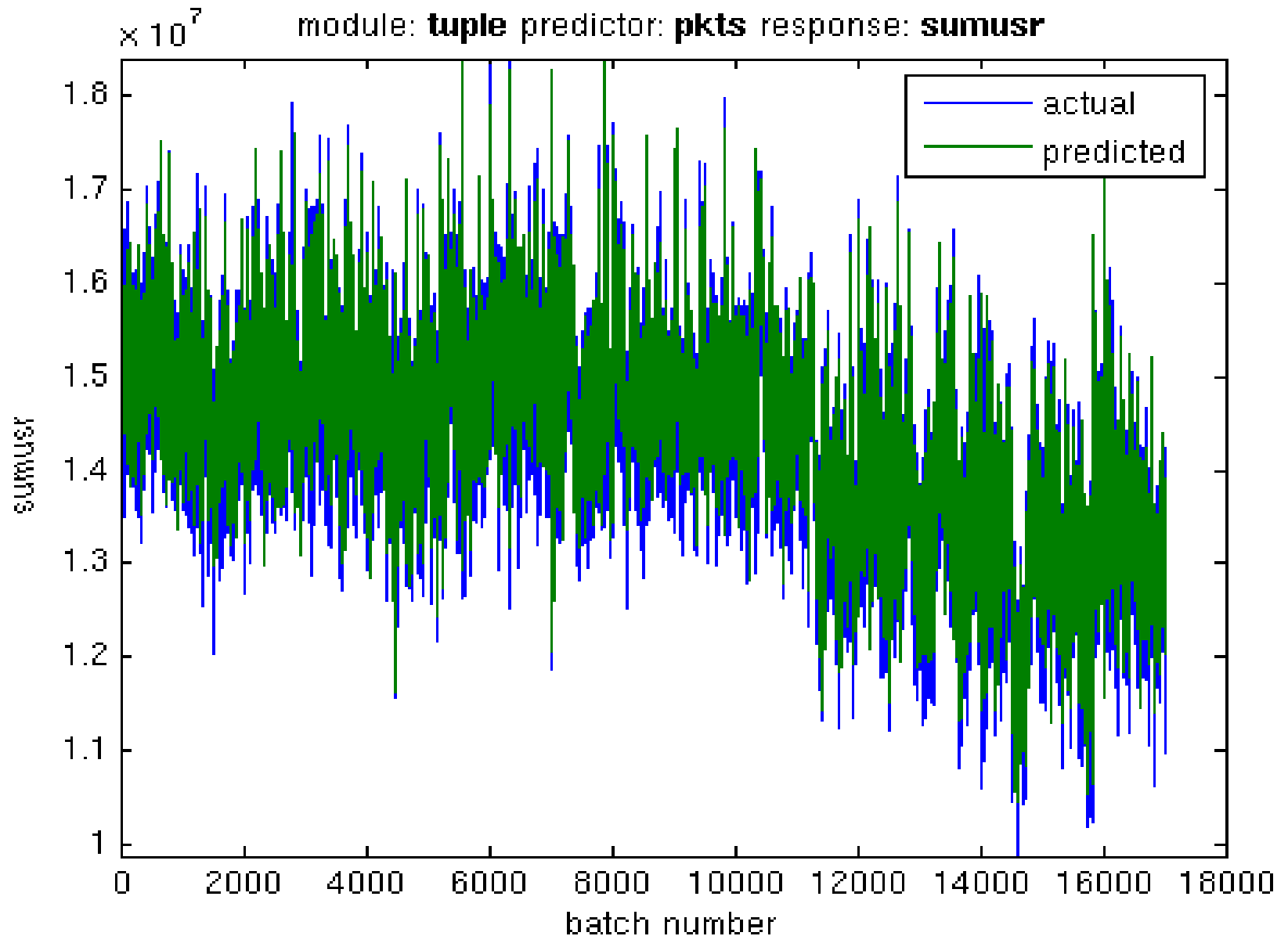
Callback cycles



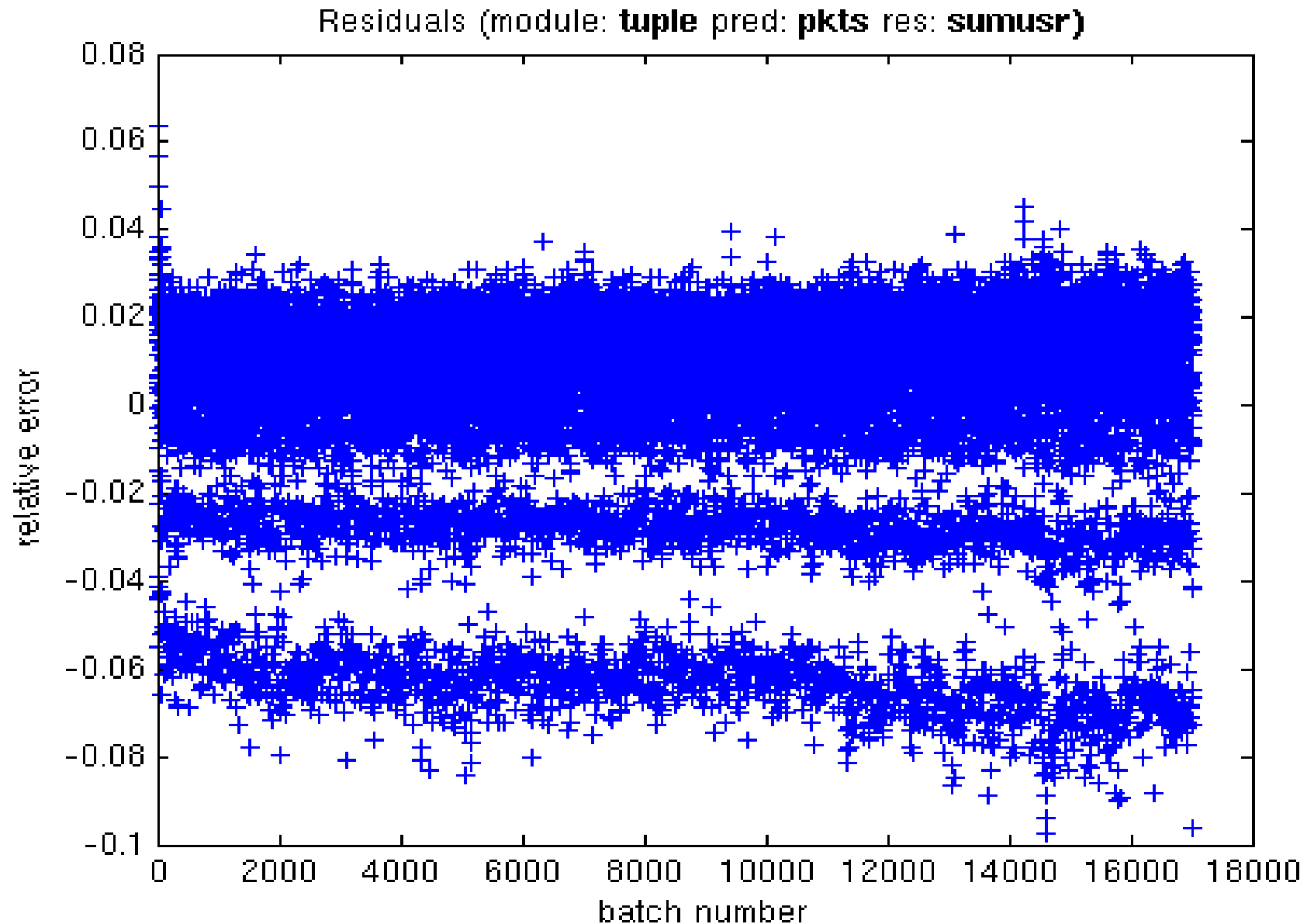
Callback cycles



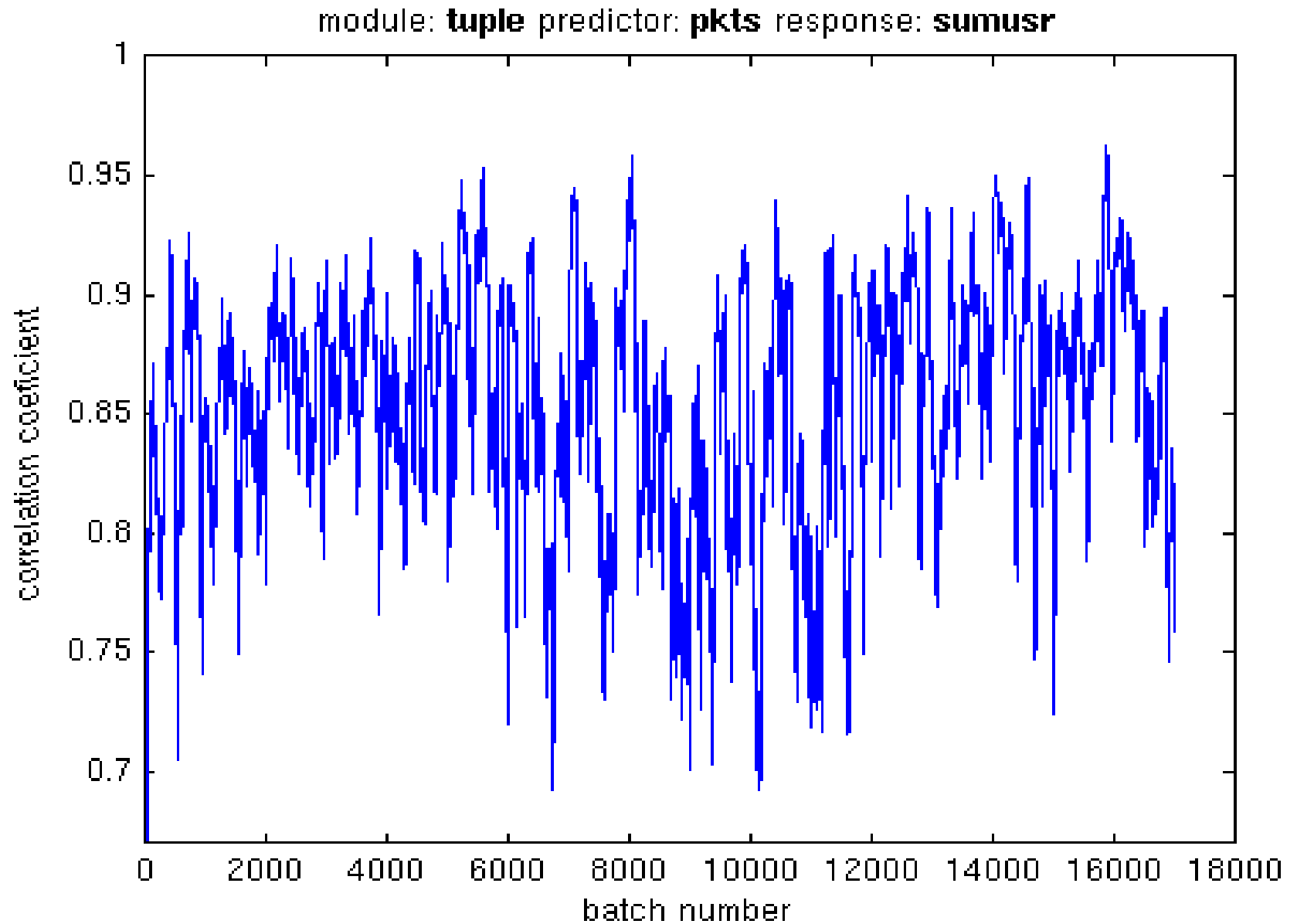
Linear regression prediction (10 sec)



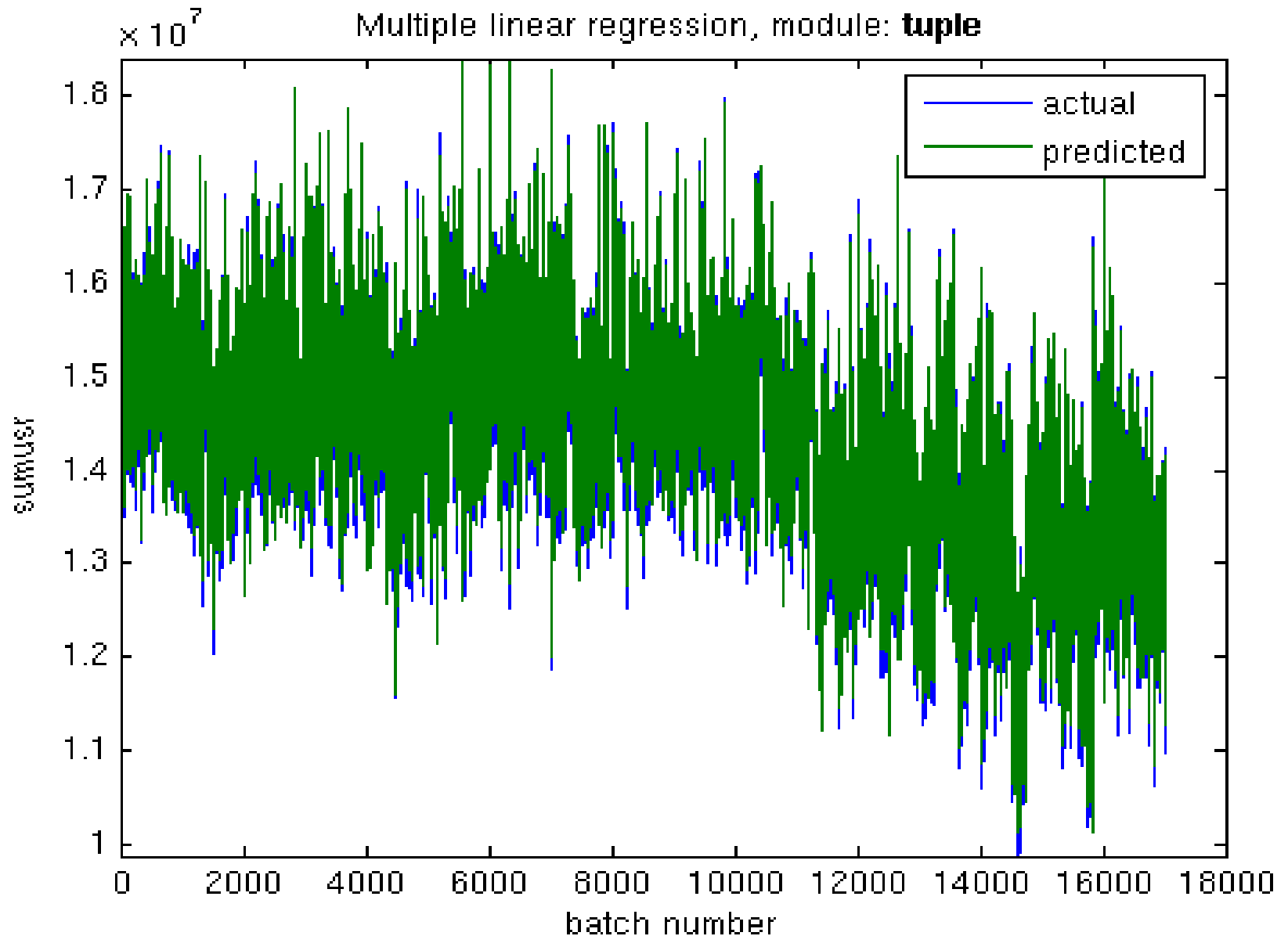
Linear regression prediction (10 sec)



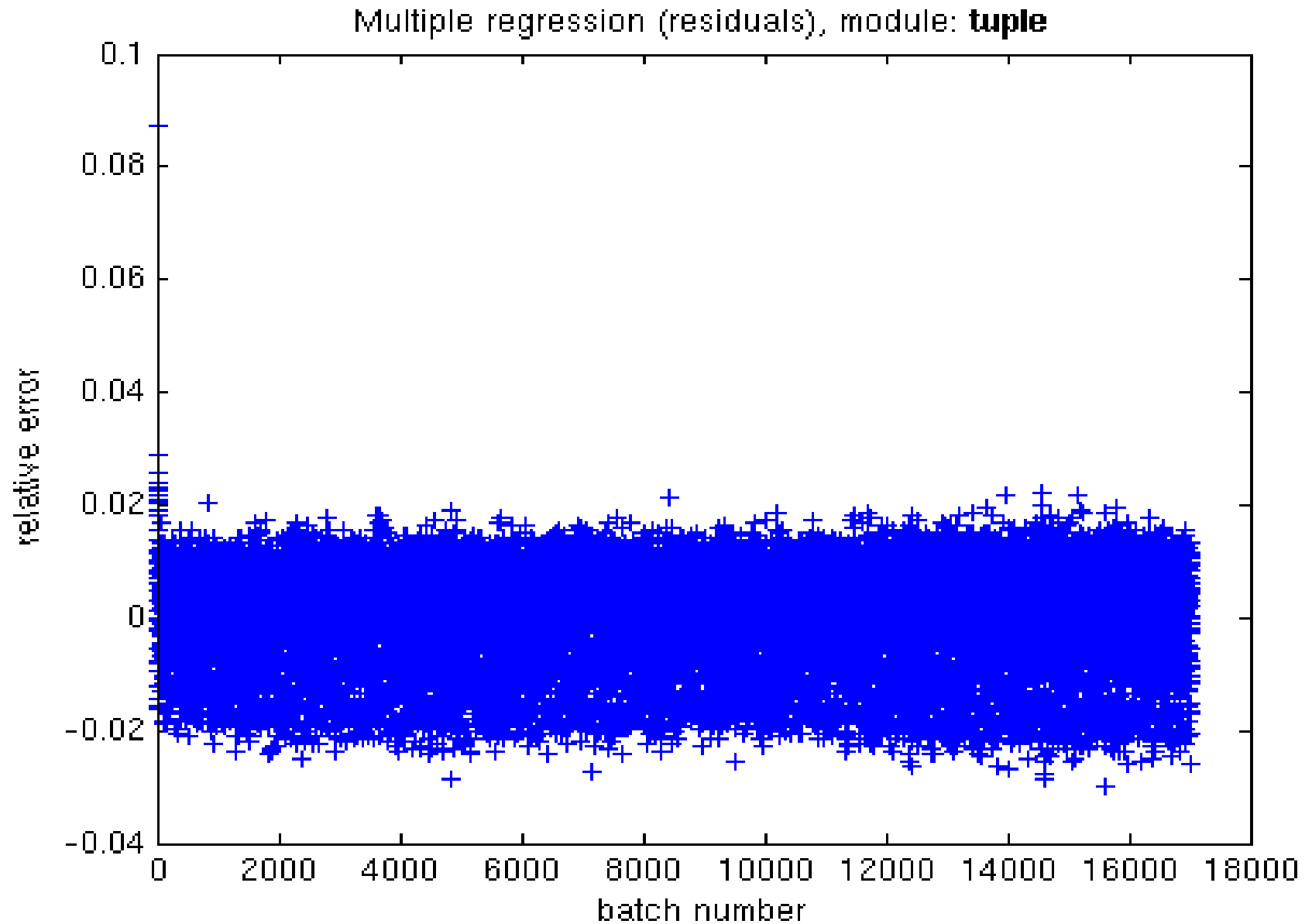
Linear regression prediction (10 sec)



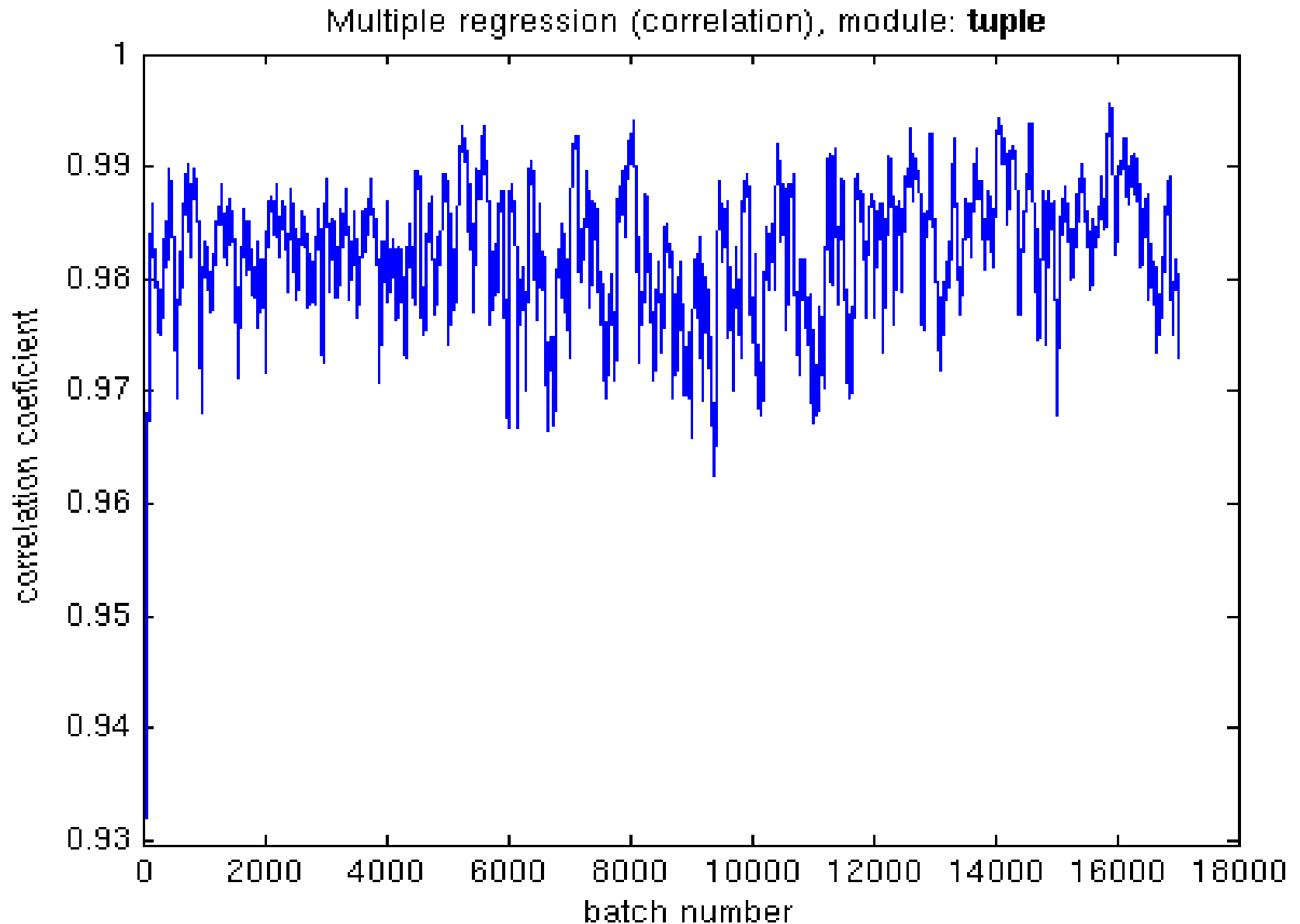
Multiple linear regression prediction (10 sec)



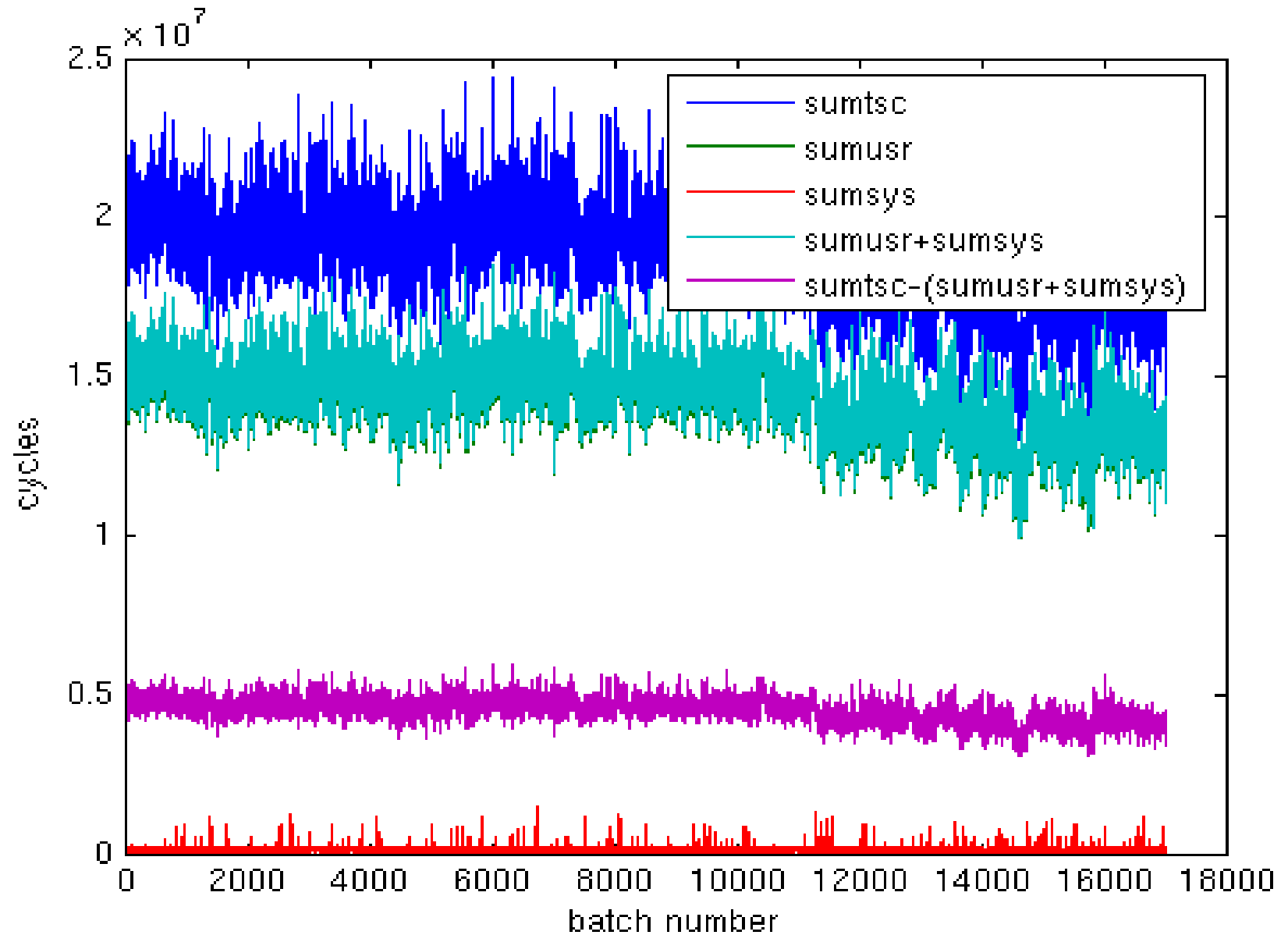
Multiple linear regression prediction (10 sec)



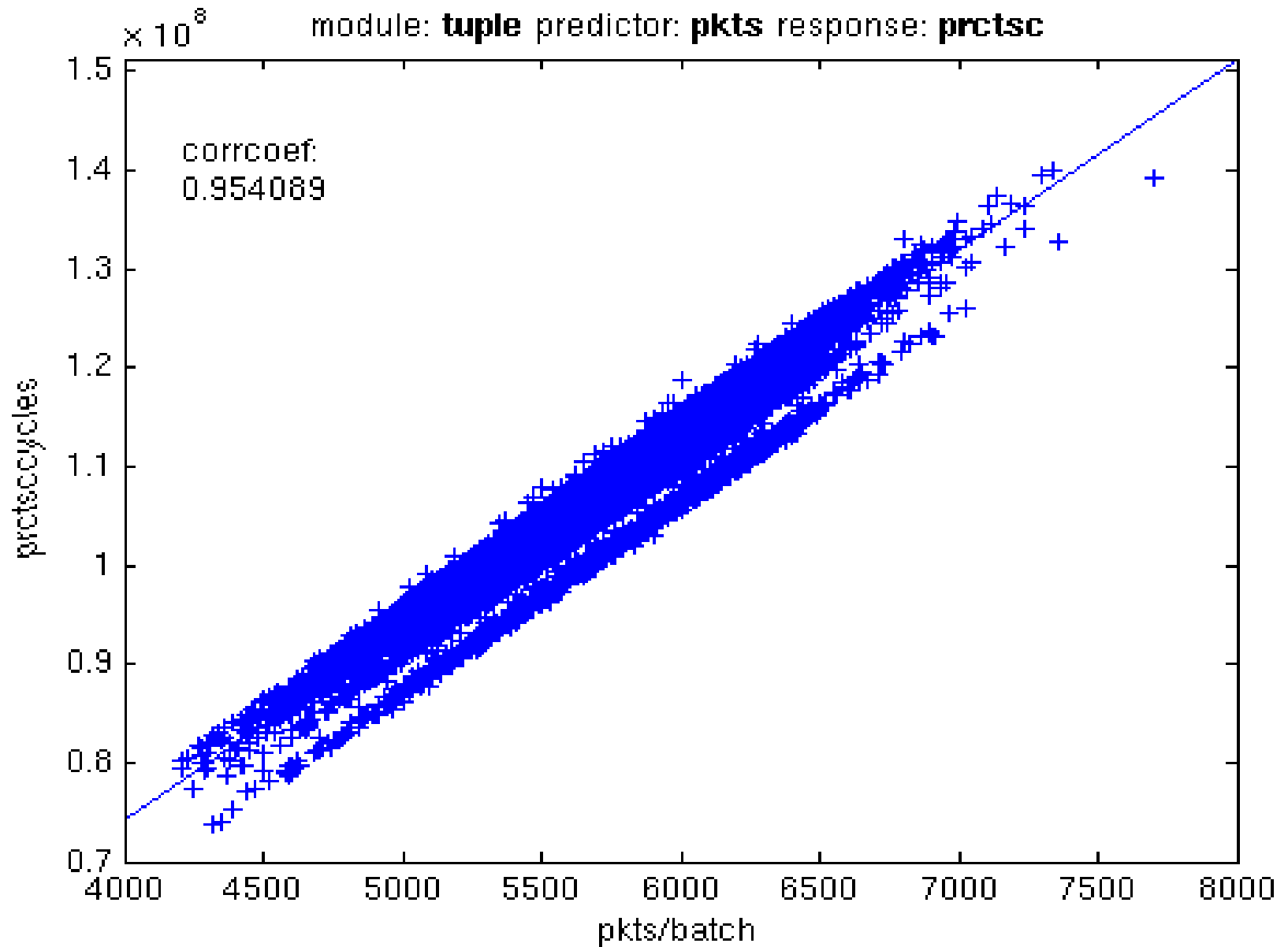
Multiple linear regression prediction (10 sec)



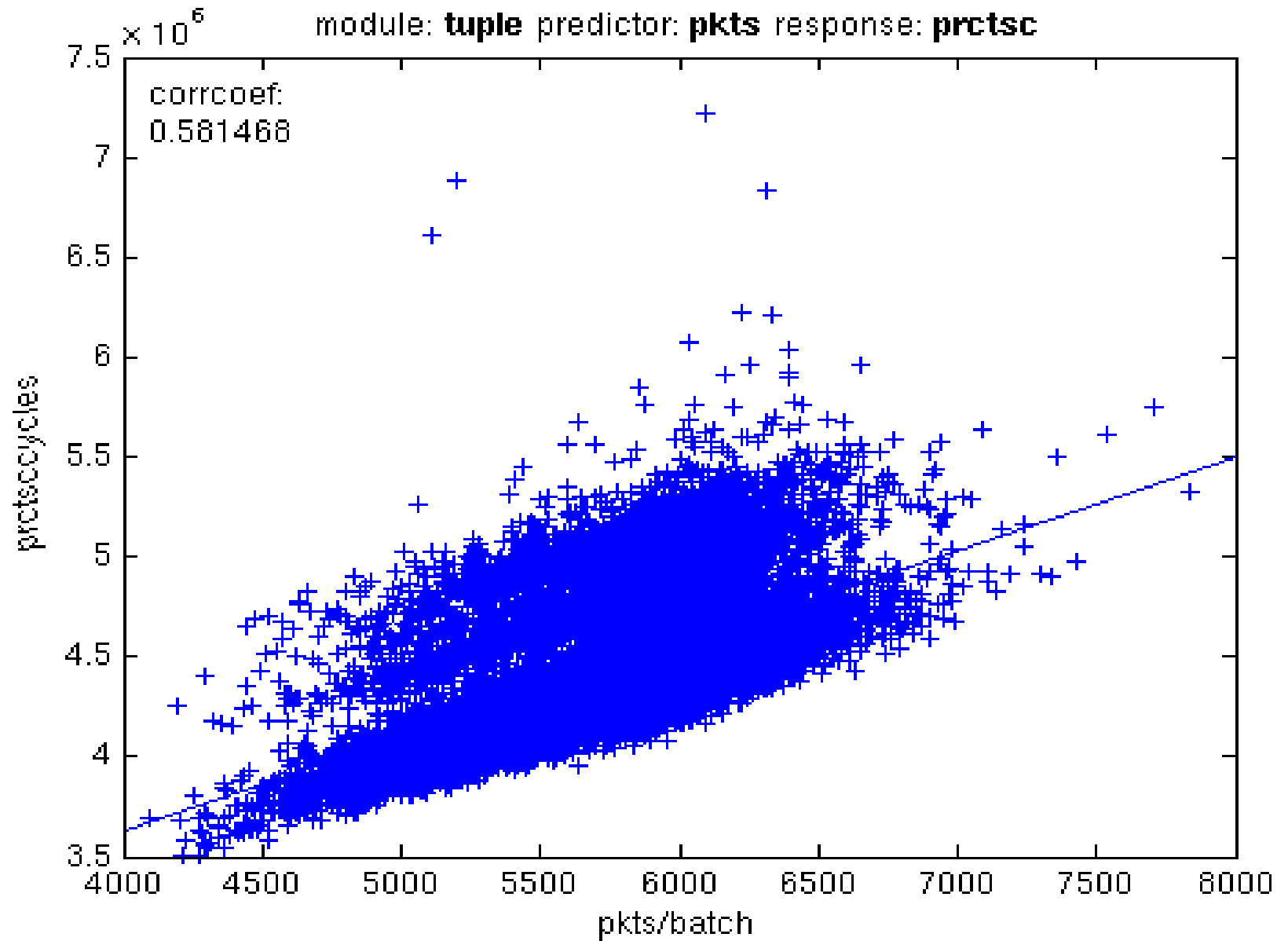
System vs. userland cycles



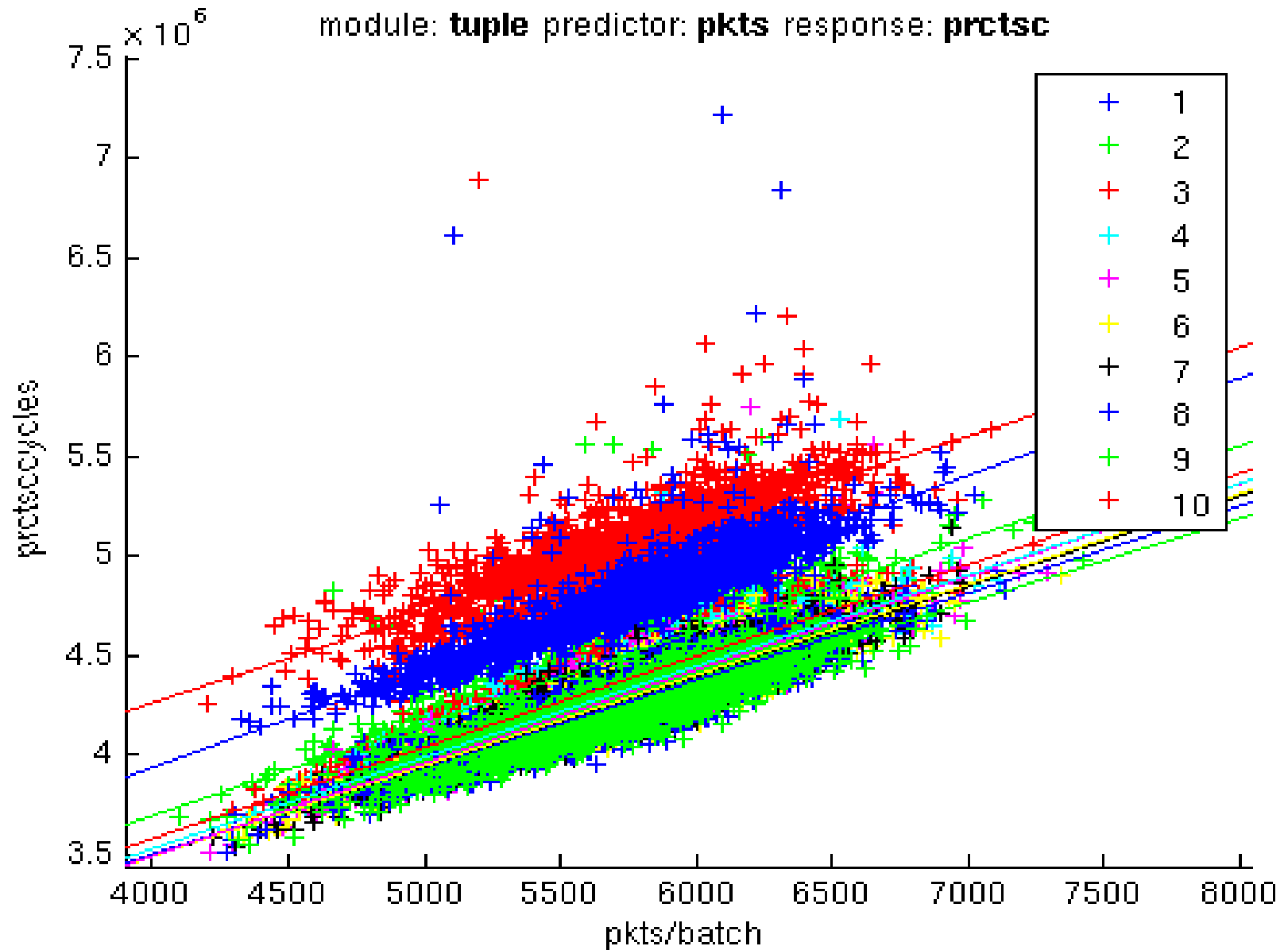
Measurement overhead/error



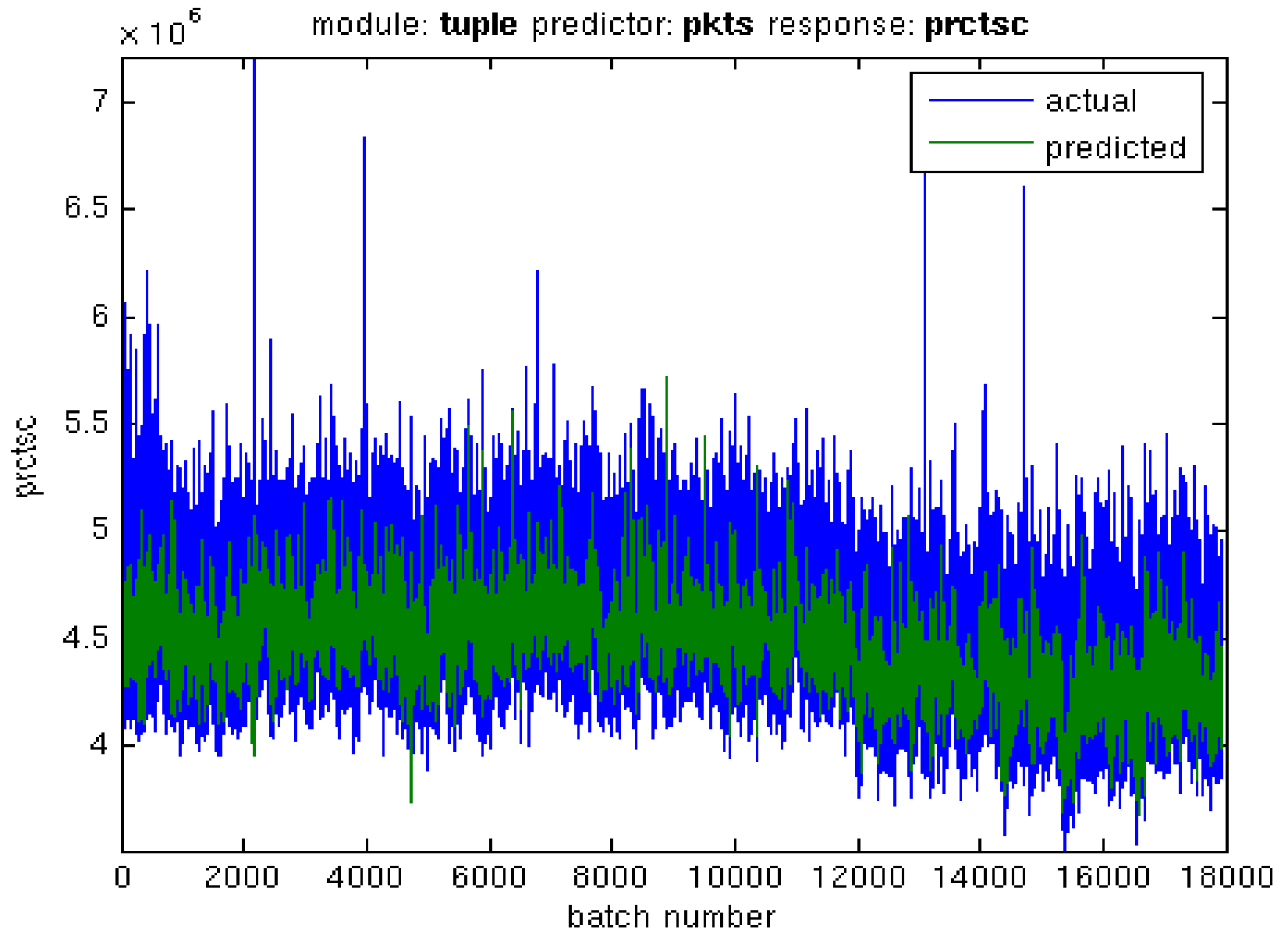
Reality



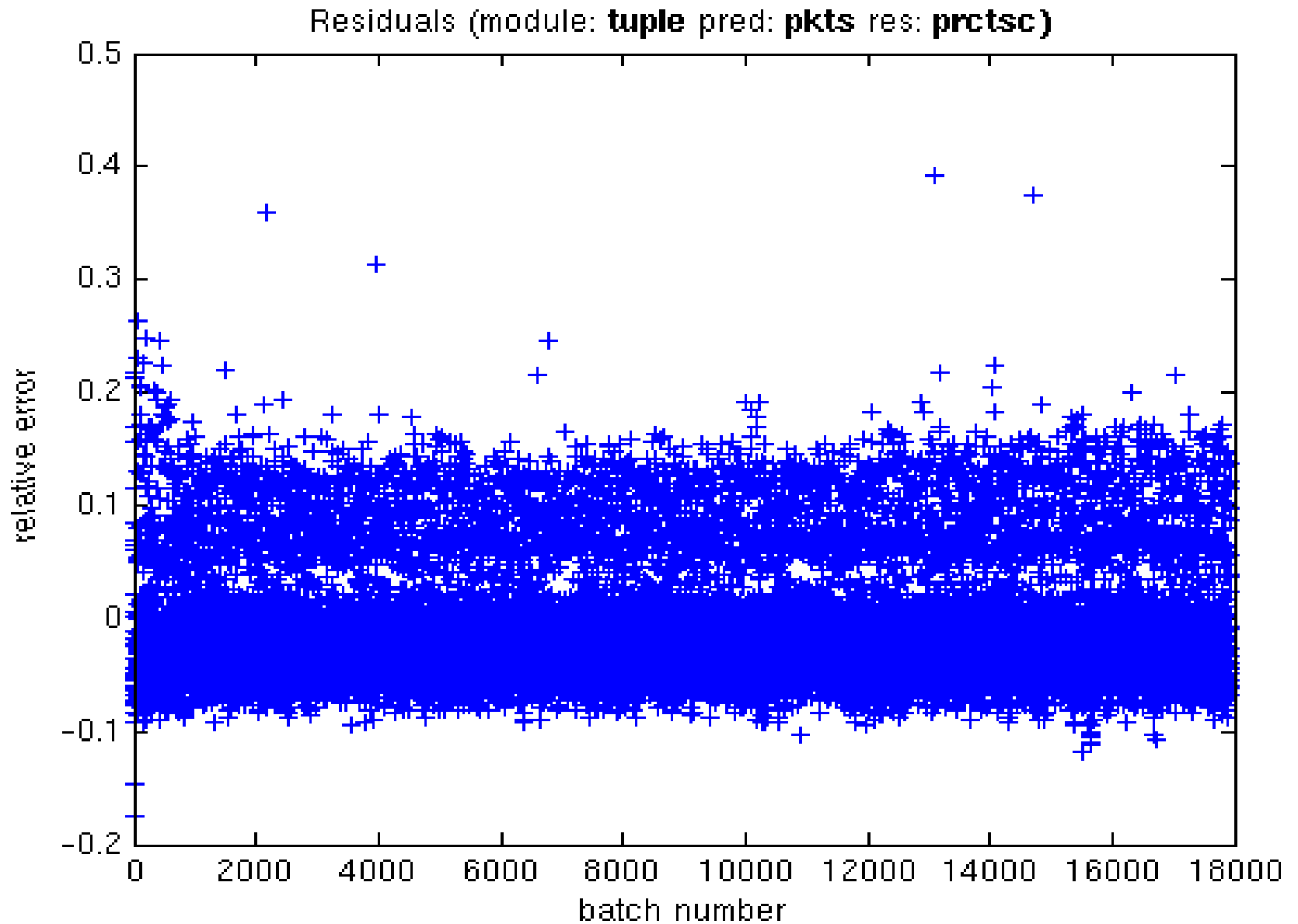
Can we predict that?



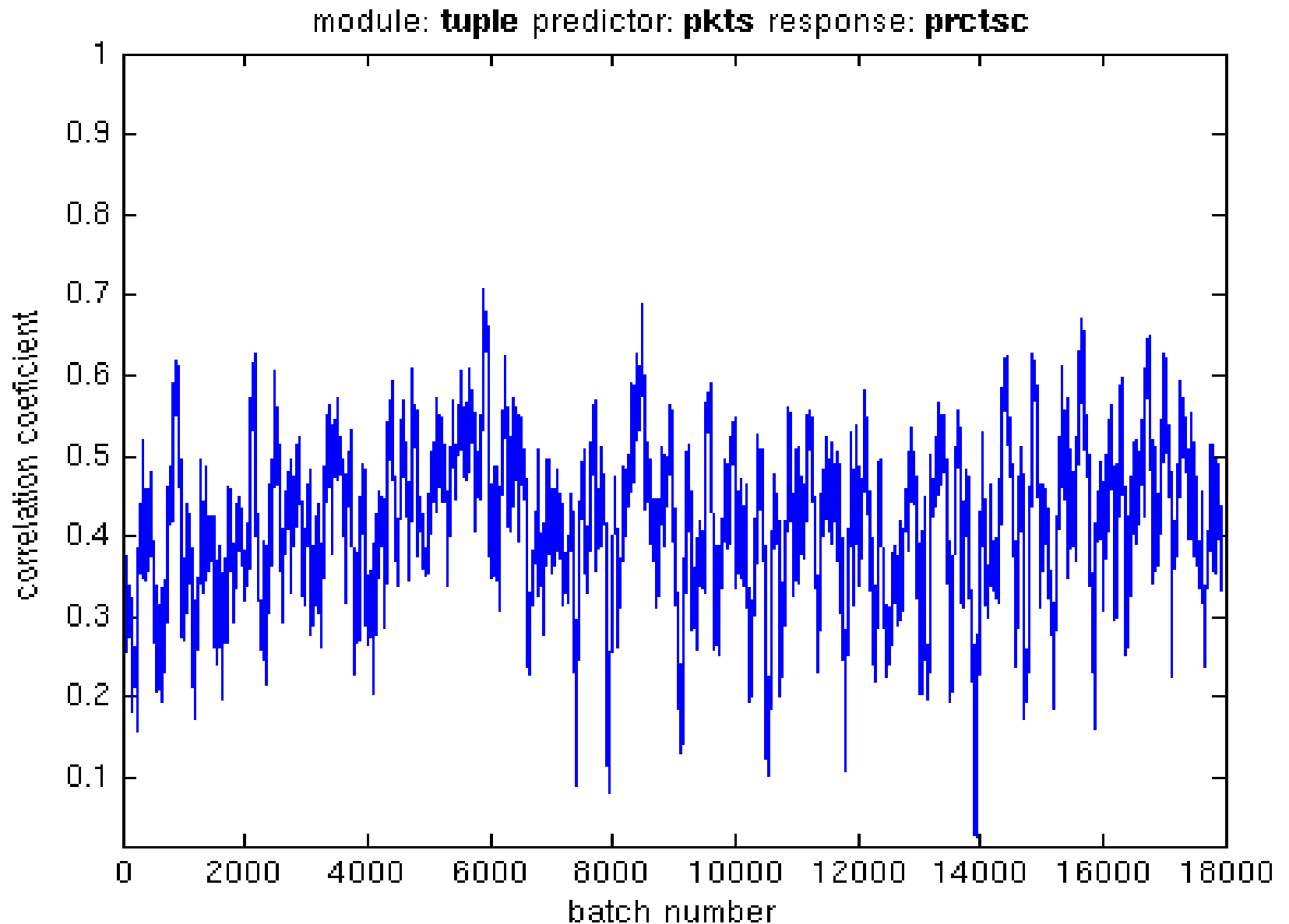
Linear regression prediction (10 sec)



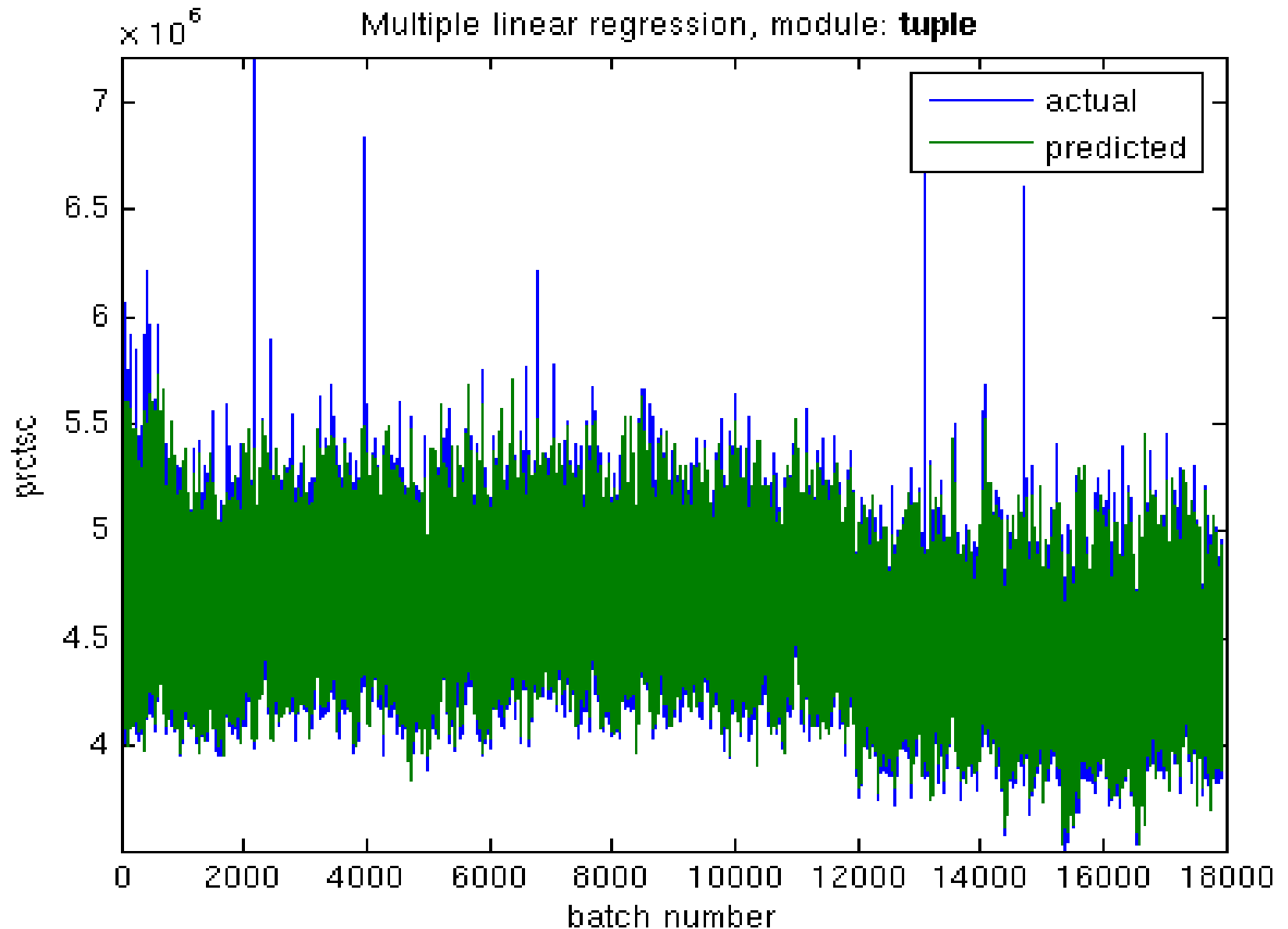
Linear regression prediction (10 sec)



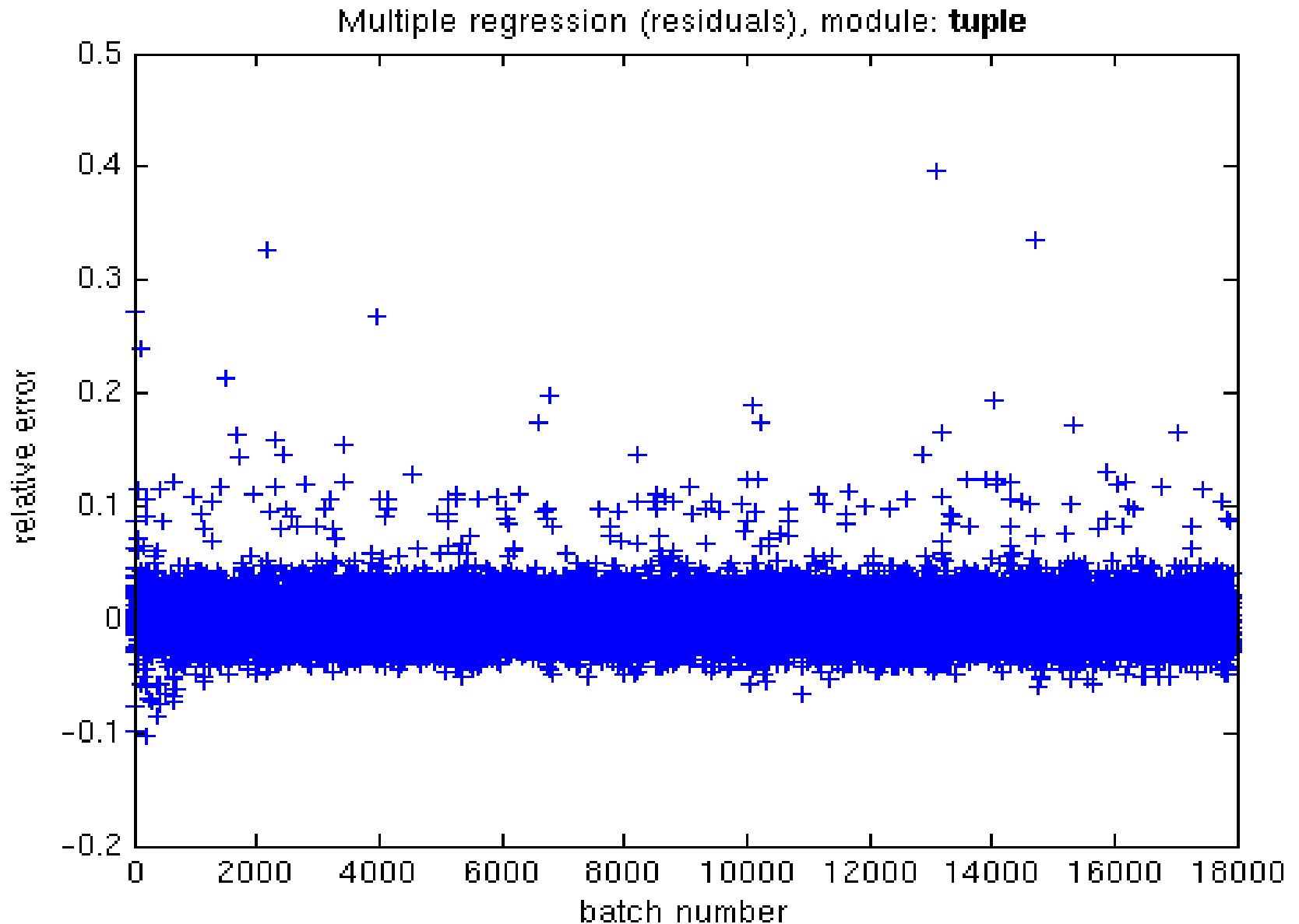
Linear regression prediction (10 sec)



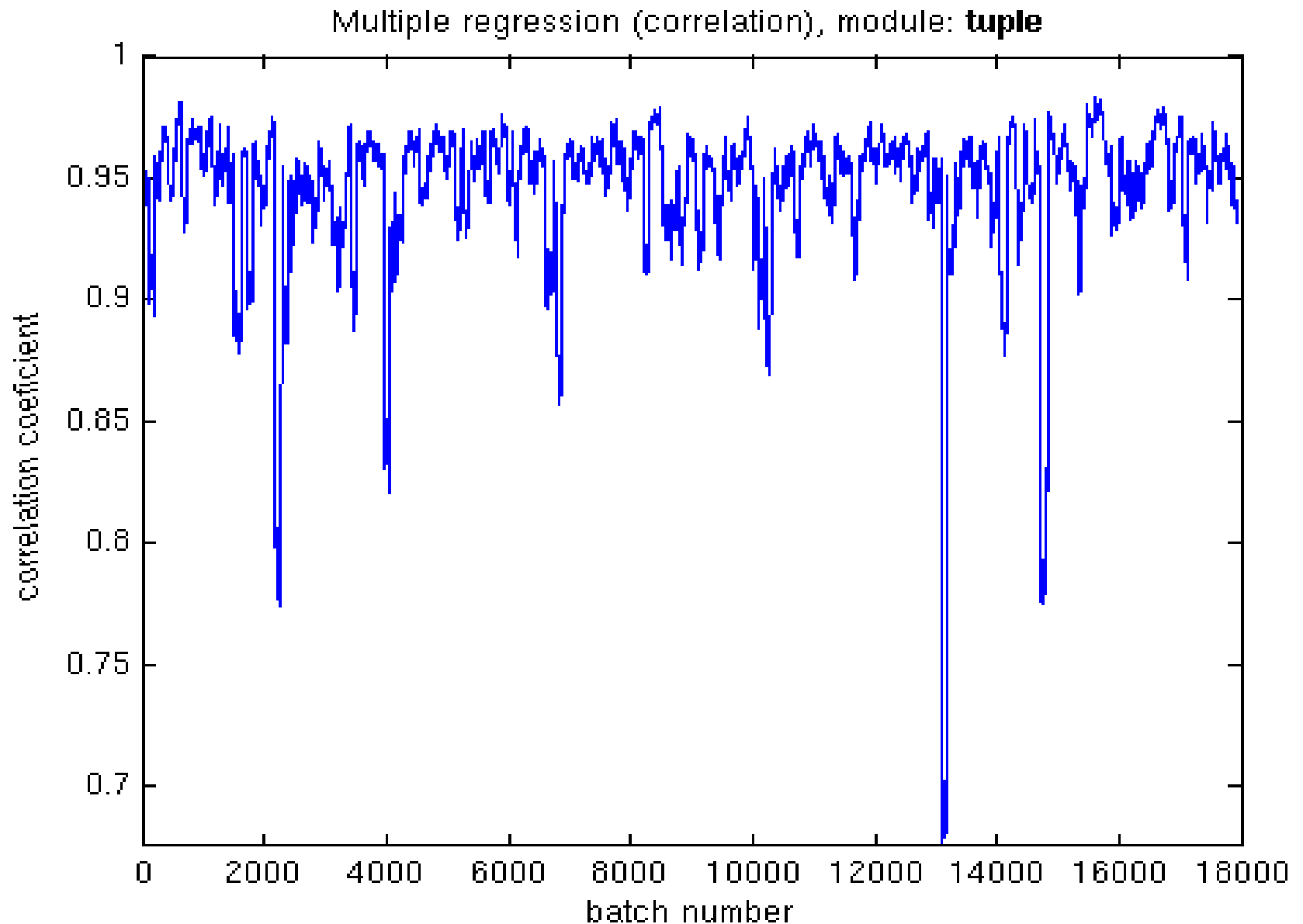
Multiple linear regression prediction (10 sec)



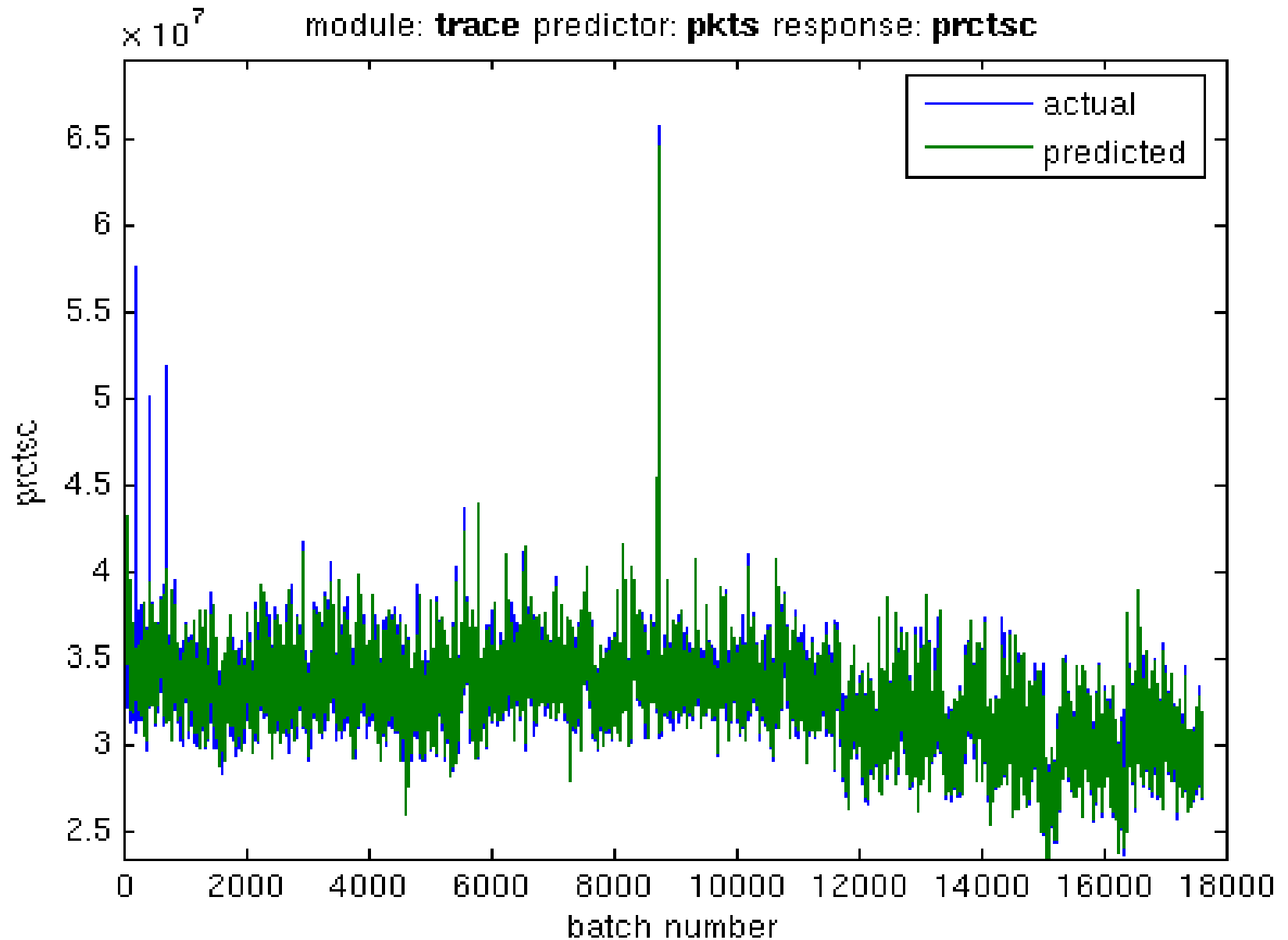
Multiple linear regression prediction (10 sec)



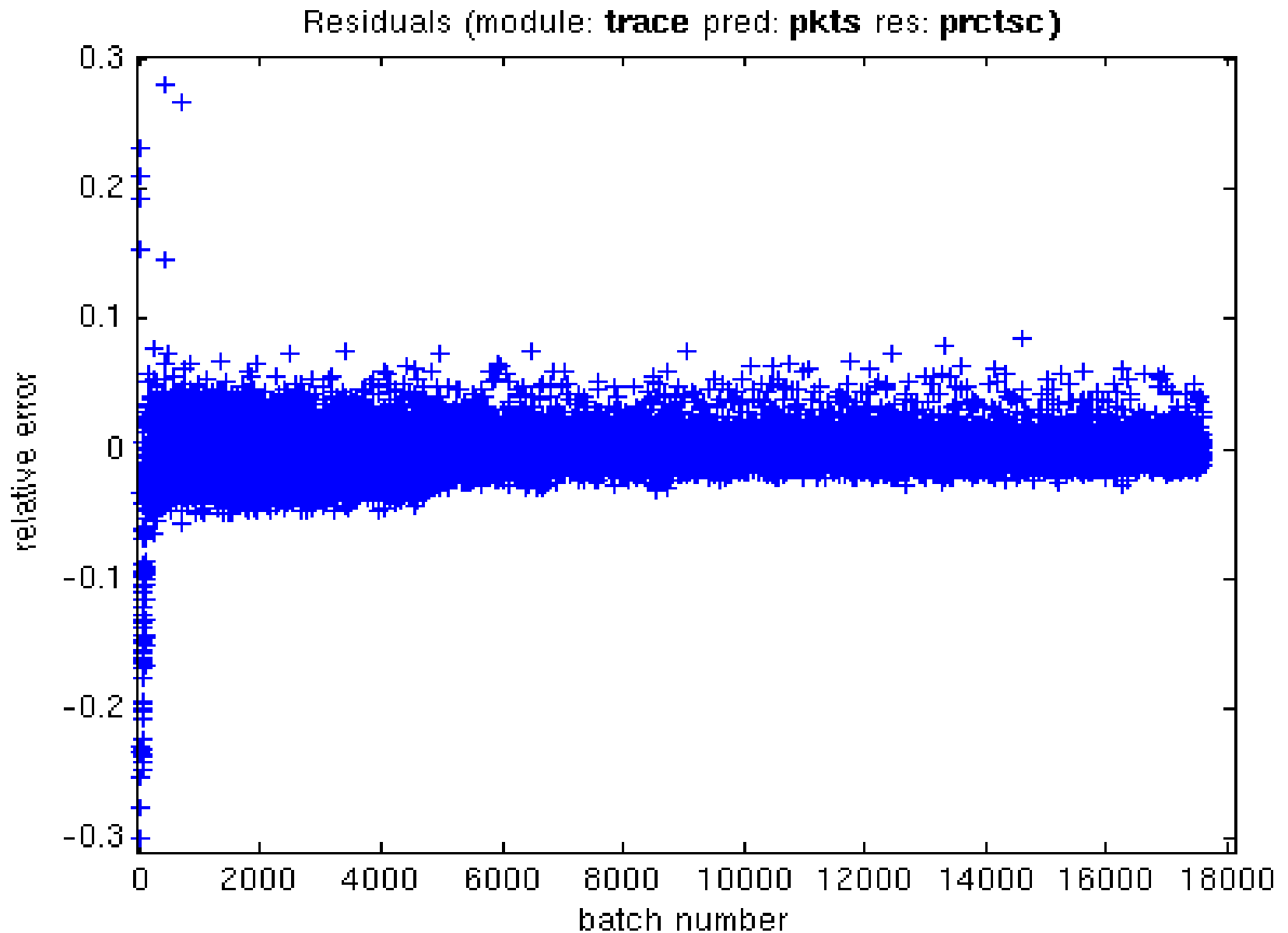
Multiple linear regression prediction (10 sec)



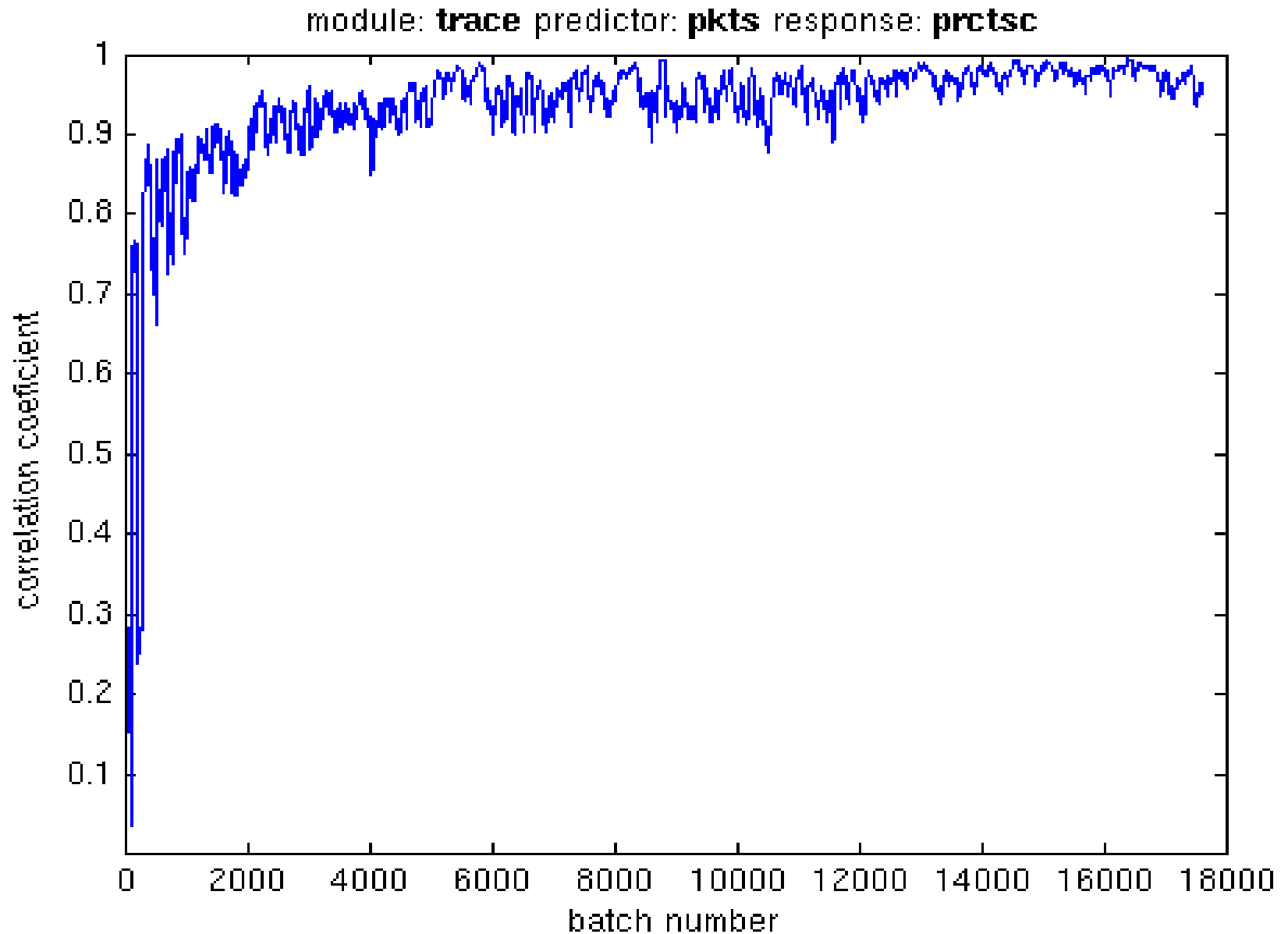
Trace: Linear regression prediction (10 sec)



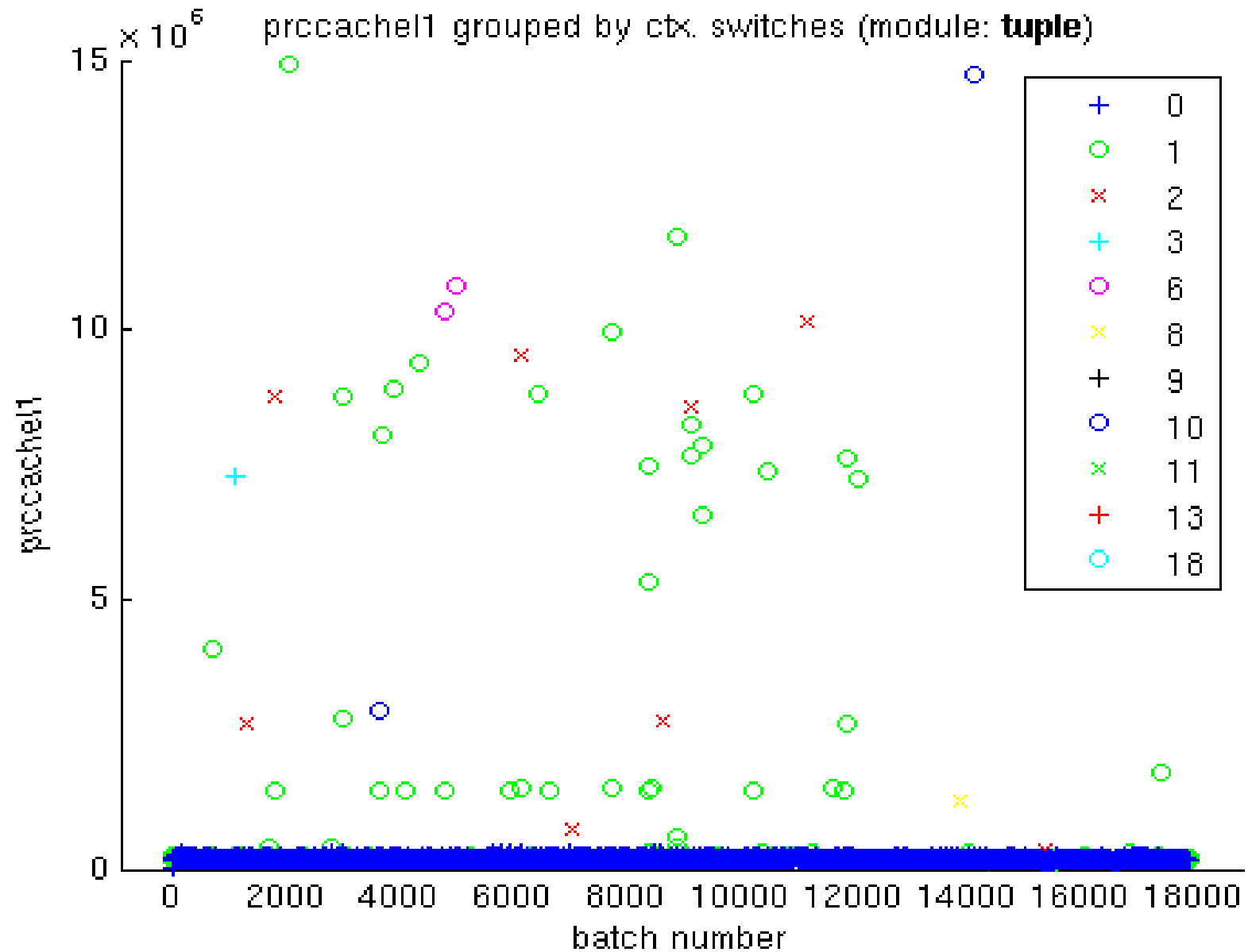
Trace: Linear regression prediction (10 sec)



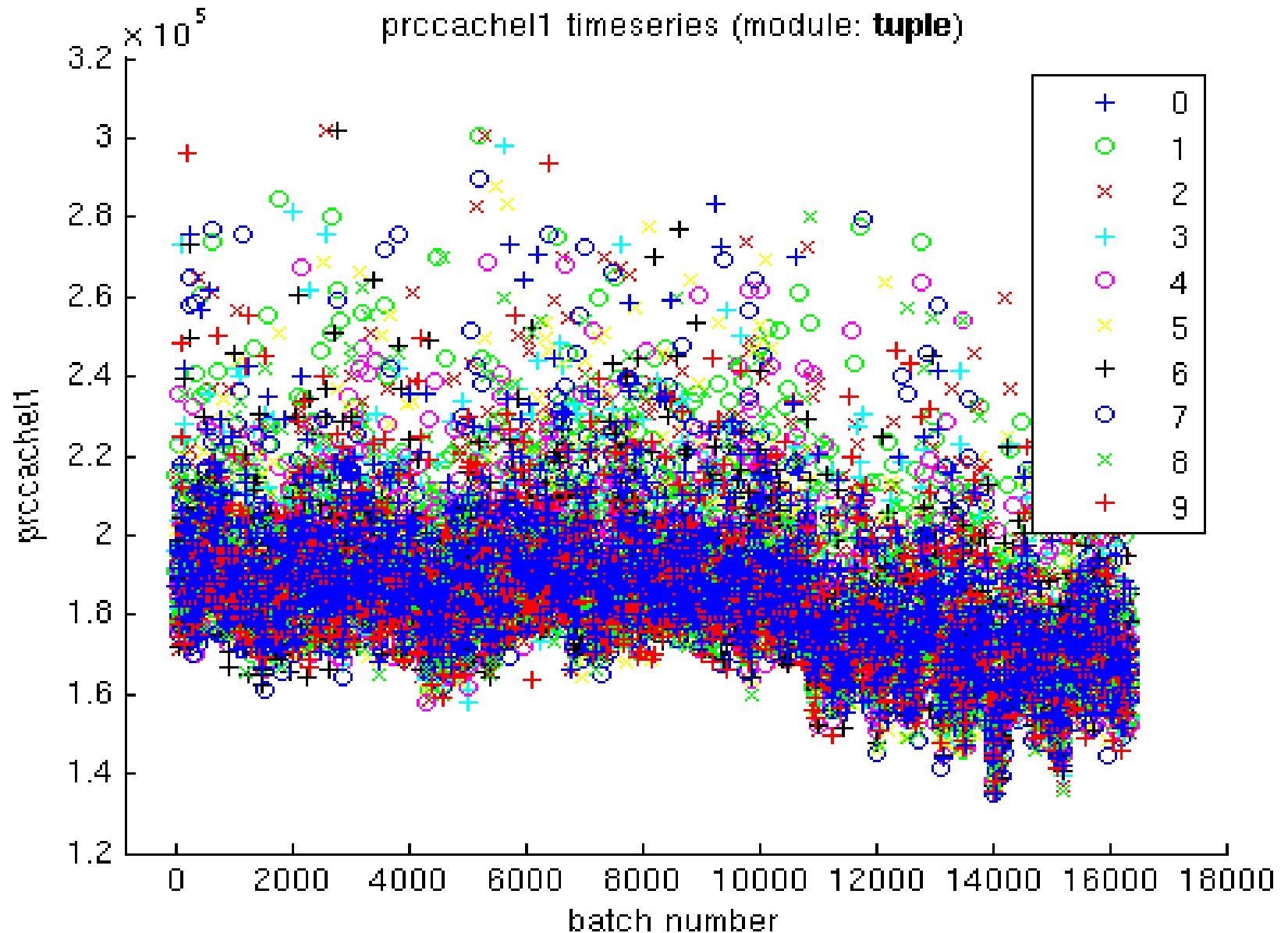
Trace: Linear regression prediction (10 sec)




Effects of context switches



Removing samples with context switches



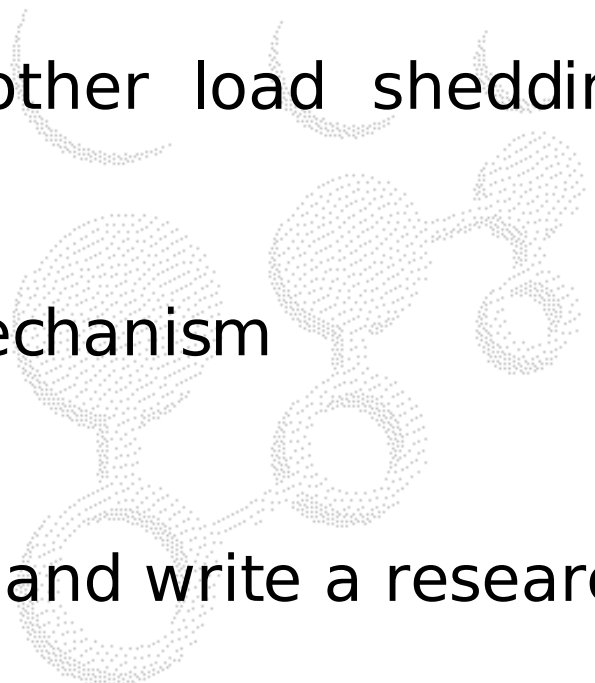
Agenda

- The scenario, challenges and objectives of SHENESYS
 - Work done and current status
 - Preliminary results
 - **Work plan**
 - **Short term work plan**
 - Other tasks
 - Equipment
 - Appendixes
 - Josep Sanjuàs: Intel performance counters
 - Diego Amores: Summary of his internship at Intel Research
- 

Work plan (deadline: August 2006)

- Implement on-line prediction in CoMo
 - Based on multiple linear regression
 - Implement a method for feature selection
- Study and improve robustness of on-line prediction mechanism in presence network anomalies and attacks
- Detect when there are no enough CPU cycles available to process a batch before the next batch arrival
 - To simplify, we might assume that capture is running alone
- Linear optimization to schedule modules in capture
 - Utility of module is given as input
 - Simple load shedding: Stop serving batches to certain modules

Work plan (deadline: August 2006)

- Analysis of more complex modules
 - e.g. SNORT, Autograph, etc.
 - Do the same for memory
 - First analysis of *export* and other load shedding mechanisms
 - Improve the profiling/logging mechanism
 - Queriable through CoMoLive!
 - Submit a paper to a conference and write a research report for project renewal
- 

Short term work plan (deadline: ~March 2006)

- Implement on-line multiple linear regression in CoMo
- Implement a fast feature selection algorithm in CoMo
 - Remove irrelevant and/or relevant attributes
 - e.g. adaptation of Fast Correlation Based Filter (FCBF)
- Modify capture to generate artificial anomalies and attacks
 - Network scans, DoS, elephant flows, etc.
- Improve resource measurement functionalities
 - Independent measurements per logical and physical CPU's
 - Check for processor switches during measures
 - Support for deactivating cache
- Minor tasks
 - Repair SNORT module, etc.

Agenda

- The scenario, challenges and objectives of SHENESYS
- Work done and current status
- Preliminary results
- Work plan
- Short term work plan
- Other tasks
- Equipment
- Appendixes
 - Josep Sanjuàs: Intel performance counters
 - Diego Amores: Summary of his internship at Intel Research



Other tasks

- Other tasks more related with the main development of CoMo
- Master Thesis' students
 - Derek Hossack working on Autograph
 - Possible topics for new Master Thesis' students
 - Anomaly detection improving the *anomaly-ewma* module
 - Identification of network applications based on heuristic techniques (port of the method already implemented in SMARTxAC)
 - Suggestions?
- Support and maintenance of CoMo nodes
 - CESCA
 - Possibly internal testing nodes at UPCnet

Equipment

- Equipment available at CCABA that can be used for Shenesisys (and CoMo Master Thesis' students):
 - Intel Xeon 3.0 GHz dual processor (giro.ccaba.upc.edu)
 - Intel Pentium IV 3.0 GHz (parellada.ccaba.upc.edu)
 - 2 x Endace 4.3 GE cards
 - 2 x SysKonnnect SK98
 - Trimble Acutime 2000 GPS receiver
 - TDS module
 - Optical splitters
- como-upc CVS: tempranillo.ccaba.upc.edu

Agenda

- The scenario, challenges and objectives of SHENESYS
- Work done and current status
- Preliminary results
- Work plan
- Short term work plan
- Other tasks
- Equipment
- Appendixes
 - Josep Sanjuà: Intel performance counters
 - Diego Amores: Summary of his internship at Intel Research



Profiling CoMo modules

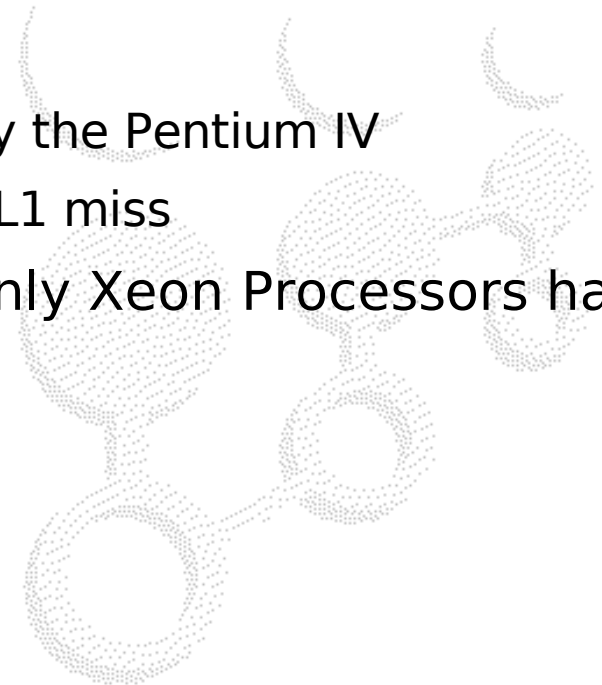
- Goal: **profiling CoMo modules' callbacks**
 - Cycles: user space, system space, total
 - Cache (L1, L2, L3): hits and misses
- Using Performance Monitoring Counters (**PMCs**) of the Pentium IV
 - Subset of its Model Specific Registers (**MSRs**) (not available on Pentium III, etc)
- Relevant documentation:
 - IA-32 Intel Architecture Software Developer's Manual
 - IA-32 Intel Architecture Optimization Reference Manual

Performance Monitoring Events

- Counting cycles:
 - TSC: **t**imesta**m**stamp **c**ounter
 - Increments on each CPU cycle
 - 64 bit (vs all other counters: 40 bit). Overflow each >10 years.
 - Architectural register, not model-specific
 - Non-halted clockticks
 - Increments on each non-halted CPU cycle (does not increment during I/O, etc)
 - Hyperthreading: can count per-logical-processor
 - Can count only system cycles, user cycles, or both

Performance Monitoring Events

- Cache misses:
 - L1:
 - no way to count misses provided by the Pentium IV
 - count instructions *replayed* due to L1 miss
 - L2, L3: can count cache misses (Only Xeon Processors have L3)



Access to PMCs

- Write operations:
 - Needed to choose what metrics are of interest (done once per CoMo execution)
 - Linux offers an interface to PMCs, so we use it: the **/dev/cpu/*/msr** virtual device
- Read operations on PMCs:
 - Read msr virtual device VS execute the rdpmc instruction
 - Read access using msr virtual device too slow, but rdpmc forbidden by linux
 - We are reading PMCs intensively
 - Wrote a **linux kernel module** that enables userland execution of the **rdpmc** (read PMC) instruction (which is not permitted by default)
- Reading the TSC:
 - rdtsc instruction, allowed from user space by default

Configuration of PMCs

- Configuration of PMCs:
 - tsc: nothing to do
 - others:
 - 1) determine event to monitor (cache misses, instructions replayed, or cycles)
 - 2) choose an appropriate event selection control register (ESCR)
 - 3) configure the ESCR to select the event to monitor
 - 3) choose an appropriate performance counter
 - 4) configure its configuration control register (CCCR) to enable counting

Preventing instruction reordering

- The Pentium IV can reorder instruction execution
 - Speculative execution of instructions (branch prediction)
 - Memory reordering
- Serializing instructions force the processor to complete all modifications to flags, registers, etc before execution of next instruction.
 - no instruction after a serializing operation can be executed before
 - no instruction before the serializing op can be executed after
- Serializing operations can impact on CPU performance
 - results of speculatively executed instructions are discarded
- The *cpuid* serializing instruction can be used to increase the accuracy of the PMCs:
 - we will not measure instructions that do not belong to callbacks
 - `cpuid; read pmcs; cpuid; call_module_callback; cpuid; read pmcs; cpuid`
 - we still need to check impact on performance

Agenda

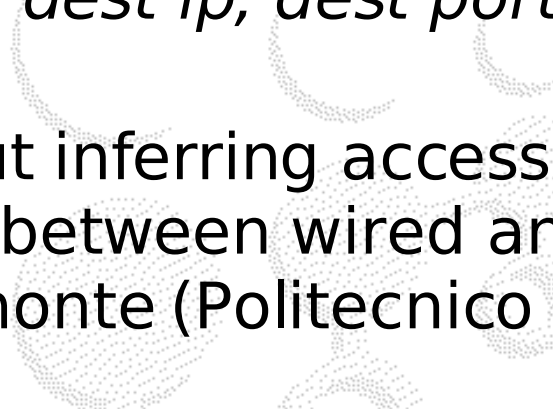
- The scenario, challenges and objectives of SHENESYS
- Work done and current status
- Preliminary results
- Work plan
- Short term work plan
- Other tasks
- Equipment
- **Appendixes**
 - Josep Sanjuàs: Intel performance counters
 - **Diego Amores: Summary of his internship at Intel Research**



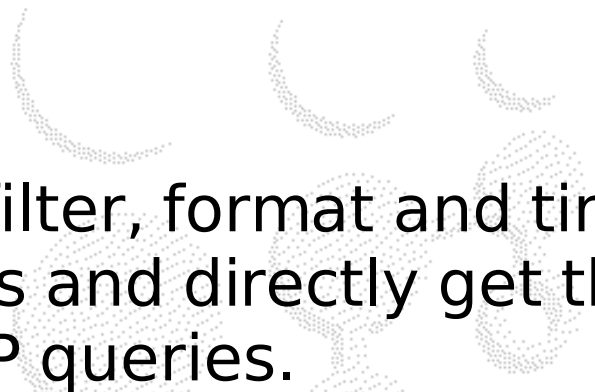
Packet filtering in CoMo

- Initial situation
 - Packet filters *compiled at runtime* and loaded as a shared library.
(undesirable because not portable to some architectures, e.g. ARM).
 - When querying, users have to write packet filters *exactly* in the same way they are configured in the system.
- New implementation of the filter parser
 - Using a CFG and Flex/Bison.
 - Filters are seen as “expression trees”.
 - No longer needed to compile filters at runtime (> portability)
 - Users only have to write *semantically equivalent* filters when querying (more flexibility).
 - Looked into more advanced packet filtering methods (PTree, Tuple Space Search), but discarded them for now.
 - Future: probably use a method similar to BPF filtering.

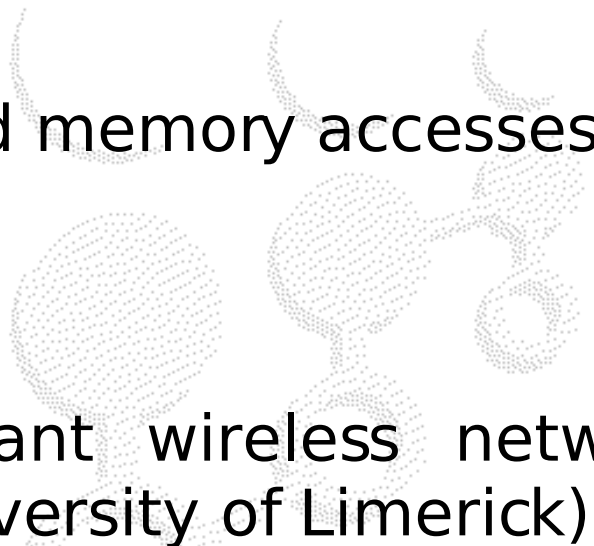
Interarrival module

- Outputs the packet timestamps for each 5-tuple (*protocol, source ip, source port, dest ip, dest port*).
 - Used in experiments at IRC about inferring access network load and distinguishing between wired and wireless traffic, by Valeria Baiamonte (Politecnico di Torino).
- 

CoMo “Inline”

- Command-line operation support for CoMo (as in other tools like tcpdump).
 - The user can specify a module, filter, format and time interval as command-line options and directly get the output, without the need of HTTP queries.
- 

Porting CoMo to ARM

- **Objective:** make CoMo work in machines with an ARM architecture (like the Crossbow Stargate for wireless monitoring).
 - **Main issue:** data structures and memory accesses must be 4-byte aligned.
 - **Future work:** Tamper resistant wireless network monitoring, by K.P.McGrath (University of Limerick).
- 

“Source” modules

- Added “source” option to queries.
- When serving a query, it is now possible to use the data stored by another module as input, through the `replay()` callback of that module.
- Makes it possible to launch queries with a time interval in the past for modules that do not store enough data themselves (e.g. `topdest` or `topports`).