Szolgáltatásbiztonságra tervezés laboratórium (VIMIM236)

Felügyeleti adatok vizuális elemzése

Mérési útmutató

Szerzők: Kocsis Imre és Salánki Ágnes

Utolsó módosítás: 2014. november 10.

Verzió: 1.3

Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

1 Bevezetés

A mérés során két, általunk menedzselt virtualizált környezetben futó alkalmazás QoS metrikájának és a futás közben rögzített platform szintű erőforrás felhasználást reprezentáló értékek között keresünk majd összefüggést.

2 Konferencia-kiszolgáló virtualizált környezetben

A mérés első fázisában egy virtualizáltan futó videokonferencia-szolgáltatás feletti méréseket fogunk vizsgálni két esetre nézve:

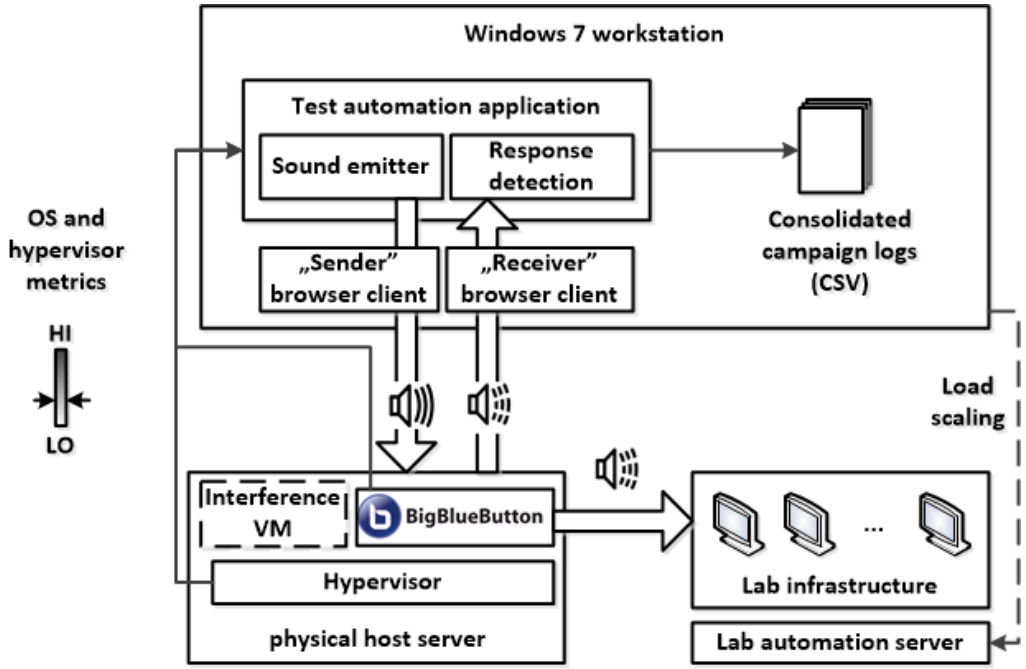
a) a kiszolgáló egy általunk menedzselt ESX infrastruktúrán fut;

b) a kiszolgáló az Amazon EC2-n fut egy bérelt virtuális gépben.

Mindkét esetben mérjük a következőket:

- Hangátvitel késleltetése
- Szabványos Linux platformszintű metrikák

Emellett az a) esetben gyűjtjük még az ESX hipervizor a hoszt erőforrásairól és a virtuális gép erőforrás-használatáról jelentett megfigyeléseit. A mérési elrendezést az a) esetben az 1. ábra szemlélteti.



1. ábra. Mérési elrendezés.

2.1 Metrikagyűjtemény

Az ESX méréseket tartalmazó bináris R adatállomány a következő oszlopokkal rendelkezik:

Oszlopnév	Leírás	Objektum	Mértékegység
timestamp	Unix epoch időbélyeg	NA	sec
Dloadavg_15m	15 perces Unix load	VM OS	NA
Dloadavg_5m	5 perces Unix load	VM OS	NA
Dloadavg_1m	1 perces Unix load	VM OS	NA

Dmemoryusage_cach	(Kernel) gyorsítótár	VM OS	byte
Dmemoryusage_free	Szabad memória	VM OS	byte
Dmemoryusage_used	Használt memória	VM OS	byte
Dnettotal_recv	Hálózaton fogadott adatok	VM OS	byte/sec
Dnettotal_send	Hálózaton küldött adatok	VM OS	byte/sec
Dprocs_new	Új folyamatok száma	VM OS	1/s
Dprocs_run	Futó folyamatok száma	VM OS	1/s
Dsystem_csw	Kontextusváltások száma	VM OS	1/s
Dsystem_int	Megszakítások száma	VM OS	1/s
Dtotalcpuusage_hiq	CPU: hardver-megszak. kezelésével töltött idő	VM OS	%
Dtotalcpuusage_idl	CPU: 'idle'	VM OS	%
Dtotalcpuusage_siq	CPU: szoftver-megszak. kezelésével töltött idő	VM OS	%
Dtotalcpuusage_sys	CPU: kernel módban futó folyamatok	VM OS	%
Dtotalcpuusage_usr	CPU: normál folyamatok felh. módban	VM OS	%
Dtotalcpuusage_wai	I/O-ra várakozás	VM OS	%
Hcpu_usage	CPU kihaszn.	ESX hoszt	% * 100
Hcpu_usageinmhz	CPU kihaszn.	ESX hoszt	% * 100
Hdisk_highestlatency	Legnagyobb SCSI parancs késleltetés (kernel + eszköz) a mintavételi intervallumban	ESX hoszt	millisec
Hdisk_readrate	Átlagos diszk olvasási ráta	ESX hoszt	kbytes/sec
Hdisk_usage	Aggregált diszk I/O ráta	ESX hoszt	kbytes/sec
Hdisk_writerate	Átlagos diszk írási ráta	ESX hoszt	kbytes/sec
Hmem_active	Aktívan használt memória („mostanában érintett" tulajdonság alapján becsülve)	ESX hoszt	kbyte
Hmem_activewrite	Aktívan írt memória	ESX hoszt	kbyte
Hmem_consumed	Összes használt memória	ESX hoszt	kbyte
Hmem_granted	A teljes VM-eknek „kiajánlott" memória (inkl. osztott) + a vSphere szolgáltatások	ESX hoszt	kbyte
Hmem_overhead	A virtuális gépek és vSphere szolg. futtatásának overhead-je	ESX hoszt	kbyte
Hmem_shared	Az ESX alkalmaz memória-deduplikációt. Ez a metrika a VM-ek „osztott" memóriamennyisége (+vSphere) szummázva.	ESX hoszt	kbyte

Hmem_sharedcommon	Osztott memóriaként funkcionáló fizikai memória. Megtakarítás: shared – sharedcommon	ESX hoszt	kbyte
Hmem_unreserved	Használatba nem vett memória	ESX hoszt	kbyte
Hmem_usage	Használatba vett memória aránya a teljeshez	ESX hoszt	% * 100
Hmem_usedbyvmkernel	A VM kernel (szigorúan vett hipervizor) által használt memória	ESX hoszt	kbyte
Hmem_zero	'0' tartalmú VM-memóriák összege ('shared' része).	ESX hoszt	kbyte
Hnet_packetstxted_vmnic1	A vmnic1 (virtuális) hálózati kártyán a mérési intervallumban forgalmazott csomagok száma	ESX hoszt	#
Qavgdelay	hangátviteli kísérletek átlagos end-to-end késleltetése (1 átviteli kísérlet: 5 „tikk" hang gyors sorozata)	szolgáltatás	millisec
Qdelaystddev	az egyes átviteli kísérleteken belüli késleltetés-szórás	szolgáltatás	millisec
Qtxfault	Átviteli kísérlet sikeres volt, ha mind az 5 „tick" felismerhetően visszaérkezett	szolgáltatás	boolean

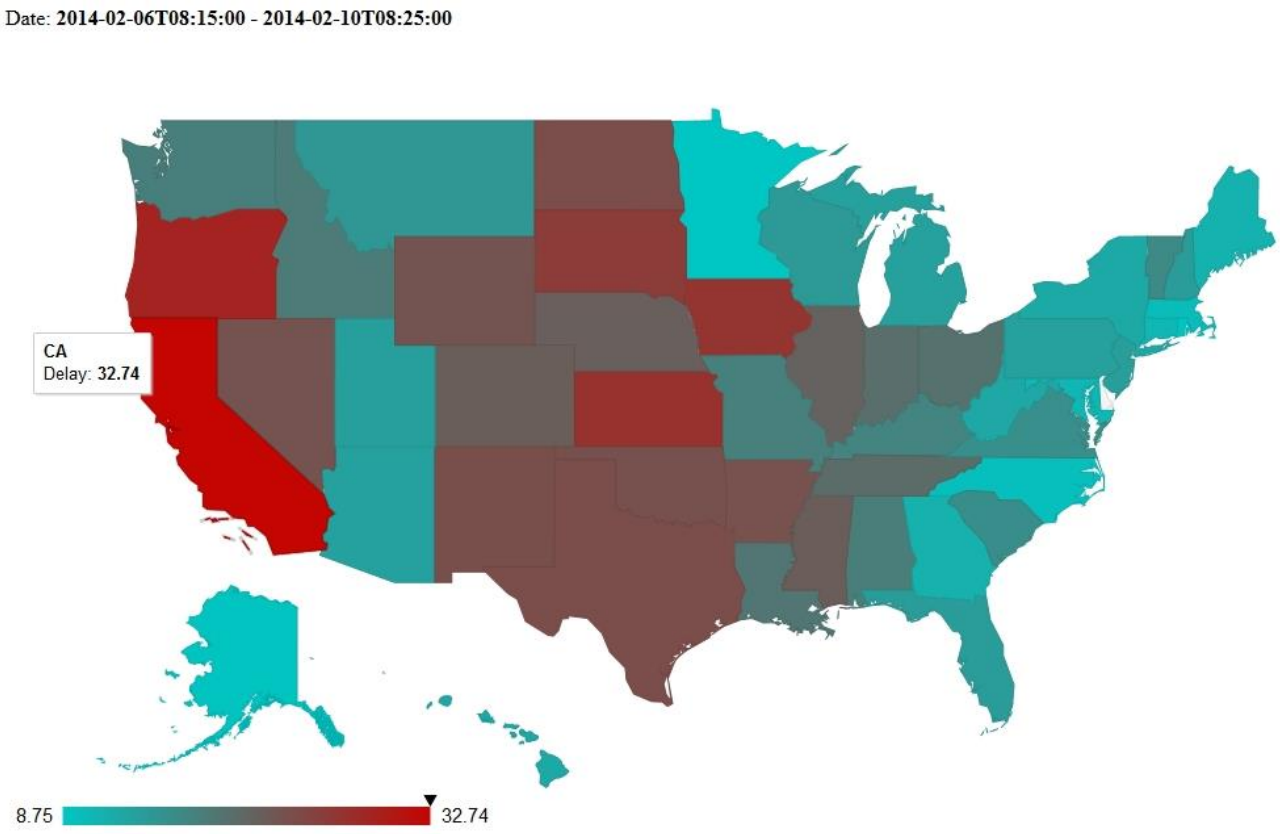
Megjegyzések: jó néhány, az ESX által jelentett metrikát kihagytunk az adatkészletből; pl. a virtuális gép szintű metrikákat teljesen (részben feleslegességük miatt, részben pedig mérési feladatokat előkészítendő). Megjegyzendő még, hogy az egyes aspektusok mintavételezése különböző frekvenciával történt.

- Linux metrikák: másodpercenként mérés.
- ESX metrikák: az ESX 20 másodpercenként frissíti a vonatkozó számlálókat. Ennél gyakrabban is lekérdezhetjük őket, de értékük ettől nem fog gyakrabban változni.
- Szolgáltatás: az átviteli kísérletek vagy az előző átvitel sikeres befejeződése („visszajött a beadott tikk-sorozat"), vagy egy timeout után indulnak.

Az adatkészletben az eltérő mintavételezési frekvenciákat kezelhettük volna úgy, hogy egy adott időpillanatban nem mért metrikák értékét `NA`-val töltjük ki, vagy interpolálhattunk is volna a megelőző és rákövetkező mérésekből. Az egyszerűség kedvéért azonban itt azt az utat követtük, hogy a sikeres/sikertelen szolgáltatás-ellenőrzés regisztrátumaira csatoltuk rá (a biztosan rendelkezésre álló) platformszintű megfigyeléseket – ezzel persze elvesztve a Linux metrikák egy jelentős részét. `NA` értékek mégis vannak az adatkeretben – ott, ahol sikertelen hangátvitel miatt nem értelmezhető az átviteli késleltetés értéke:

3 OpenStack mérések

A mérés második fázisában egy OpenStack környezetben végrehajtott kísérlet során rögzített adatokon egy teljes adatelemzési munkafolyamat első lépéseit hajtjuk majd végre, ebben a fejezetben a környezet és az alkalmazás logikai topológiáját ismertetjük. Az alkalmazás a nagy méretű adatok elemzésnél leggyakrabban használt flightdata[1] adatsort dolgozza fel, amely 2014-ben a január-február hónapok során az Amerikából induló repülőjáratok utazási információit tartalmazza. Az Apache Storm keretrendszerben futó programunk konkrétan egy adott időablakban indított járatok átlagos késését számolja ki és jeleníti meg percekben, ehhez az adatfolyamok elemzése R-ben és Pythonban történik, az adattárolást a Redis biztosítja, a térképes megjelenítést pedig a Map elnevezésű GoogleChart támogatja.



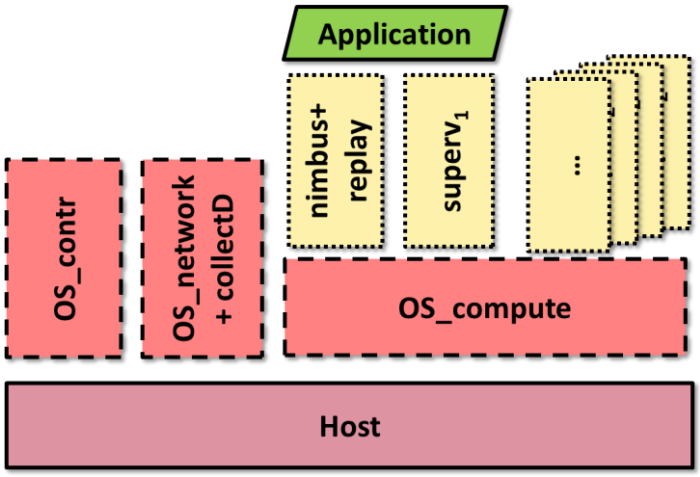
Egy, az alkalmazás futása során készített képernyőkép

3.1 Kísérleti környezet

Virtualizált környezetünket egyetlen fizikai hoszton futó három virtuális gép, az OS_Compute, az OS_contr és az OS_network alkotta. Az OS_Compute csomópont felelős a virtuális gépek futtatásáért, az OS_network gépen a collectd alkalmazás segítségével történik a rendszer monitorozása, míg a rendszerfunkciók ellenőrzését az OS_contr modul végzi.

Az alkalmazás futásához két VM szükséges, a superv1 csomóponton történik a késési átlagok kiszámítása és megjelenítése, a nimbus+replay az előzőleg letöltött adatok visszajátszásáért, illetve a menedzsment funkciók ellátásért felelős. A környezet egy harmadik, superv2 névvel ellátott virtuális gépet is tartalmaz, az ezen futó szolgáltatást ismeretlennek tekintjük.

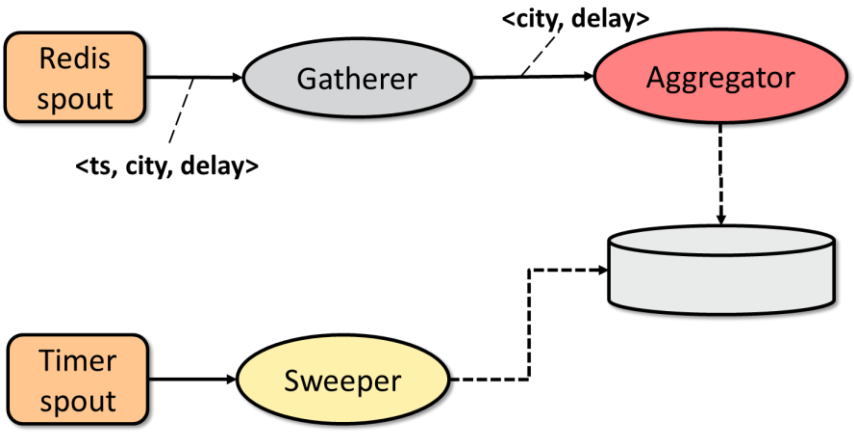
[1] http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236&DB_Short_Name=On-Time



Az OpenStack mérési környezet

3.2 A Storm alkalmazás

Az Apache Storm egy elosztott, hibatűrő, adatfolyamok valós idejű feldolgozását is lehetővé tévő keretrendszer. A projekt alapvetően nyílt forráskódú volt a fejlesztő BackType nevű cég 2011-es felvásárlásáig. Az alkalmazások Storm környezetben való futtatásához egy „spout"-okból és „bolt"-okból álló logikai topológiára való leképezése szükséges, ezek között történik folyamokban az adatcsere.



A flightdata adatsort feldolgozó alkalmazás topológiája

Alkalmazásunk két spoutot tartalmaz, az első (Redis spout) a Redisben tárolt kulcs-érték párok kiolvasásért, míg a második (Timer spout) az időablak karbantartásáért felelős.

A Gatherer komponens implementálja a szűrő logikát, az Aggregator csomópontban történik az átlagolás, a Sweeper pedig a Timerből érkező sweeper hatására a megfelelő adatokat ki- és beteszi a feldolgozó bufferbe.

3.3 Metrikagyűjtemény

Monitoring metrics	
timestamp	The number of seconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970, not counting leap seconds.
machine	The hostname of the machine on which the metric was recorded.
core	On a multi-core hardware, different values can be measured among cores. This value is the numeric ID of the core.
disk_partition	Linux based systems identify the disks by a short series of characters, and partitions with integers starting with one. Ie. sda designates a whole disk, while sda2 is the second partition of sda.

OS level metrics	
cpu_idle	Jiffy is the unit of cpu effort, the time spent on a job. A jiffy usually means 10ms of cpu time. Cpu metrics are the number of jiffies spent on a job since boot. Cpu_idle counts the jiffies the cpu spent without job.
cpu_user	The number of jiffies consumed by normal priority processes in user mode. Generally all processes run in user mode.
cpu_nice	In most Unix-like systems, a value called “nice” can be given to processes to modify it's priority. Cpu_nice is the number of jiffies spent on user mode processes with modified nice value. Nice is usually used to decrease priority, so this metric counts the processing effort used by low priority tasks.
cpu_wait	Time spent on waiting for I/O requests.
cpu_system	The number of jiffies spent in kernel mode.
cpu_softirq	The number of jiffies spent on serving software interrupts.
cpu_interrupt	The number of jiffies spent on serving hardware interrupts.
mem_free	The amount of memory left unused.
mem_cached	The size of the kernel page cache. All file I/O is performed through this cache. The cache is shrinked or purged only when the needed, so the cache size can grow to several gigabytes and can use up all the memory without causing any problem.
mem_buffered	The size of the buffer for block I/O operations. These records are short-lived, so the buffer size has low importance nowadays.
mem_used	The amount of memory used by processes.
disk_octets_[write|read]	The number of octets read from and written to the disk since boot.
disk_ops_[write|read]	The number of I/O operations requested since boot.
disk_merged_[write|read]	The number of operations, that could be merged into other, already queued operations since boot. Merging operations reduces the load on the disk, thus improving performance.
disk_time_[write|read]	The average time an I/O-operation took to complete. Its basically the sum of time spent in I/O operations devided by the number of operations. While this value is very useful
timestamp	The number of seconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970, not counting leap seconds.
Application level metrics[2][3]	
Acked	number of fully processed tuples
Capacity	derived metric (deduce from Executed and ExecuteLatency)
Emitted	the number of calling emit() method
Executed	the number of calling execute() method
ExecuteLatency	time spent in execute for a tuple
Executors	number of executors
Failed	number of failed tuples (A tuple is considered failed when its tree of messages fails to be fully processed within a specified timeout)
ProcessLatency:	time until tuple is acked
Tasks	number of tasks
Transferred	the number of transferred tuple. (not neccessary equals to the number of emitted tuples due to different stream grouping)

[2] http://storm.incubator.apache.org/documentation/Guaranteeing-message-processing.html

[3] http://storm.incubator.apache.org/documentation/Understanding-the-parallelism-of-a-Storm-topology.html

4 Mérési feladatok

Felkészülésként indítsa el a mérésvezető által meghatározott virtuális gépet és másolja fel a megadott, a mérés során majd használt adatokat.

4.1 ESX adatok – interaktív, vizuális vizsgálat

0. Indítsa el a `Mondrian`-t és:
    a. ha még nem tette meg, olvassa el a http://www.rosuda.org/Mondrian/ oldal 'Quickstart' és 'Concepts' részét,
    b. a Help menüből nyissa meg az alkalmazás „Reference Card"-ját a billentyűkombinációkkal ismerkedés céljából!
1. Nyissa meg a `BBB_ESX_v2.csv`-t! (Ez az azonos nevű .RData `Mondrian` által közvetlenül beolvasható csv-képe.)
2. Nyugodtan töltsön el némi időt a plot-típusok kipróbálásával.
3. Vizsgálja meg a szolgáltatásminőséget leíró metrikákat! Mi történt a kísérlet során a szolgáltatásunkkal?
4. Határozza meg, hogy mérnöki szempontból mely megfigyelt platform-metrikák nem voltak praktikusan állandóak a kísérlet során!
5. Az így leszűkített platform-metrika készlet és szolgáltatásminőségi metrikák között keressen összefüggéseket – a nagyszámú ábra kinyitása elkerülése érdekében párhuzamos koordináták segítségével!
    a. Mit jelent, ha egy párhuzamos koordináta diagramon két tengely közötti vonalak egy pontban metszik egymást? Miért?
    b. Talál-e olyan párt a szolgáltatásminőségi metrikák valamelyikéhez, mellyel ez teljesül?
    c. Fogalmazzon meg sejtés(eke)t a szolgáltatásminőség és a platformmetrikák közötti statisztikai összefüggés(ek)re!
6. Vizsgálatai alapján milyen mérnöki következtetést tud levonni: mi okozhatta az üzemzavart?
7. Hány virtuális gép futott a kísérlet során a hoszton?

4.2 ESX adatok – R vizsgálatok

0. Indítsa el az `RStudio`-t és ismerkedjen meg az alkalmazással! Javasolt a segédletben bemutatott R példák használata.
1. Olvassa be az ESX kísérletek eredményét tartalmazó .RData állományt!
2. Csatoljon egy relatív (0-val kezdődő) időbélyeg oszlopot az adatkerethez! (A létrehozott kódokat javasolt R parancsállomány formájában elkészíteni és tárolni.)
3. Kategorikus változóként hozzon létre egy 'kísérleti fázis' jelentésű oszlopot, mely a szolgáltatás működésében látható 'fázisokkal' annotálja a megfigyelés-sorokat!
4. Hozza létre a szolgáltatásminőségi metrikák idősorait pl. mint ggplot2 szórásdiagramokat (vagy vonaldiagramokat), a 'fázisok' szerint színezve! Segítség: a csoportszín ('`colour`') ugyanúgy az esztétikai jellemzők része `ggplot2`-ben, mint pl. az x koordináta megadása.
5. Illesszen lineáris modellt a válaszidő 'lecsengő' részére és vizualizálja is azt! ('Normál' plot-ok esetén már létrehozott ábrára az `abline` függvénnyel, `ggplot2` plotoknál a `stat_abline` függvénnyel lehet „egyenest húzni".) Mennyire jó az illesztés?
6. És ha a válaszidő logaritmusát vesszük?

7. Határozza meg erőforrástípus-csoportonként a változók korrelációs mátrixait! Milyen meglátások olvashatók ki ezekből?

4.3 OpenStack adatok

A mérés második fázisában egy teljes elemző munkafolyamatot végzünk majd el, a magas szintű metrikák elemzésétől a futtató virtuális gépek erőforrás-felhasználásán át egészen a környezet analíziséig.

4.3.1 Magas szintű metrikák vizsgálata

Nyissa meg az 1_high_level_only_long.csv állományt Mondrianban, majd válaszolja meg az alábbi kérdéseket!

1. Mik a tipikus értékei az egyes komponensek (aggregator, gatherer, sweeper) késleltetési eloszlásának?
2. Mi a tipikus értékkészlete az egyes csomópontok terhelési metrikáinak? Hogyan néz ki ennek az időbeli változása?
3. Mi a maximális kapacitás, amit az egyes csomópontok elértek és időben hogyan történt ennek a változása? Melyek azok az időbélyegek, amik későbbi vizsgálatokat igényelhetnek az eddigiek alapján? (Esetleg ezeket jegyezze is föl.)
4. Figyelembe véve a terhelés és a QoS metrikák alakulását, határozza meg a lehetséges működési tartományokat!

4.3.2 Virtuális gép szintű metrikák vizsgálata

1. A fent meghatározott időbélyegek segítségével készítsen markert (*Options → Derive Variable from*), határozza meg az egyes kategóriák számosságát!
2. Az egyes CPU metrikák vizualizálásával fogalmazza meg, milyen jelenségek figyelhetőek meg az erőforrások változásában a markerrel megjelölt időpontokban!
3. Végezze el a vizsgálatot a többi erőforrás metrikára is, adjon lehetséges mérnöki interpretációt az adatokban látott jelenségek okára!

4.3.3 Környezet vizsgálata

5. Vizsgálja meg a környezet (OS_compute, superv2 csomópontok) erőforrásmetrikáit és állítson fel hipotéziseket a jelenségek okait illetően!
6. A Mondrian zoom funkciója segítségével elemezze az átmeneteket jellemző változásokat, majd az időbeli szekvenciák alapján szűkítse a hipotézisteret!