VYUŽITÍ DATABÁZÍ V RÁMCI FYZIKÁLNÍHO EXPERIMENTU COMPASS

Lucie Fleková, Vladimír Jarý, Tomáš Liška, Miroslav Virius
České vysoké učení technické v Praze, Fakulta jaderná a fyzikálně inženýrská
Lucie.Flekova@cern.ch, Vladimir.Jary@cern.ch, Tomas.Liska@cern.ch, Miroslav.Virius@cern.ch

ABSTRAKT:

Příspěvek se zabývá problematikou softwarového zabezpečení sběru dat ve fyzikálním experimentu COMPASS v CERN a specifiky zpracování dat ve fyzice vysokých energií. Prezentuje návrh nové databázové architektury pro tento experiment.

ABSTRACT:

COMPASS is a collider experiment running on the Super Proton Synchrotron at European Organization for Nuclear Research (CERN). This paper analyses the data acquisition system used by this experiment with focus on the database infrastructure. At first, the COMPASS experiment is briefly introduced, and then the data acquisition system is described in more details. COMPASS stores event headers containing information about detector configuration, beam parameters, and other logs into MySQL databases. Some tables contain several gigabytes of data, so this paper also addresses database optimization techniques, such as table indexing or table partitioning. Finally, a new database design is proposed. To obtain high availability and reliability, the proposal includes MySQL replication, load balancing, monitoring, and regular backups.

KLÍČOVÁ SLOVA:

COMPASS, MySQL, sběr dat, optimalizace dotazů

ÚVOD

Experimenty ve fyzice vysokých energií kladou vysoké nároky nejen na špičkové technologie použité v detektorech a urychlovačích, ale také na počítačové systémy použité při sběru a analýze dat. V tomto článku se zaměřujeme na použití databází v rámci fyzikálního experimentu COMPASS. Nejprve ve stručnosti představíme tento experiment, pak rámcově popíšeme systém pro sběr dat a role databází v tomto systému. V průběhu experimentu vzniká velké množství dat, proto značnou pozornost věnujeme technikám optimalizace SQL dotazů. V závěru navrhujeme aktualizaci stávající databázové architektury.

STRUČNÉ PŘEDSTAVENÍ EXPERIMENTU

COMPASS je experiment z oboru fyziky elementárních částic, který probíhá na urychlovači SPS (Super Proton Synchrotron) v CERN od r. 1992. Jeho název je zkratkou slov *COmmon Muon and Proton Apparatus for Structure and Spectroscopy*, tedy společné zařízení pro strukturu a spektroskopii mionů a protonů.

Jedná se o experiment s pevným terčem ostřelovaným svazkem urychlených částic. V terči dochází ke vzájemnému působení mezi atomy terče a dopadajícími částicemi a k produkci sekundárních částic, které jsou detekovány rozsáhlým systémem detektorů.

Poznamenejme, že ve skutečnosti jde o dva experimenty – mionový a hadronový – spojené pod jednu hlavičku. Experimentu se účastní mezinárodní skupina vědců z více než 25 výzkumných ústavů a univerzit.

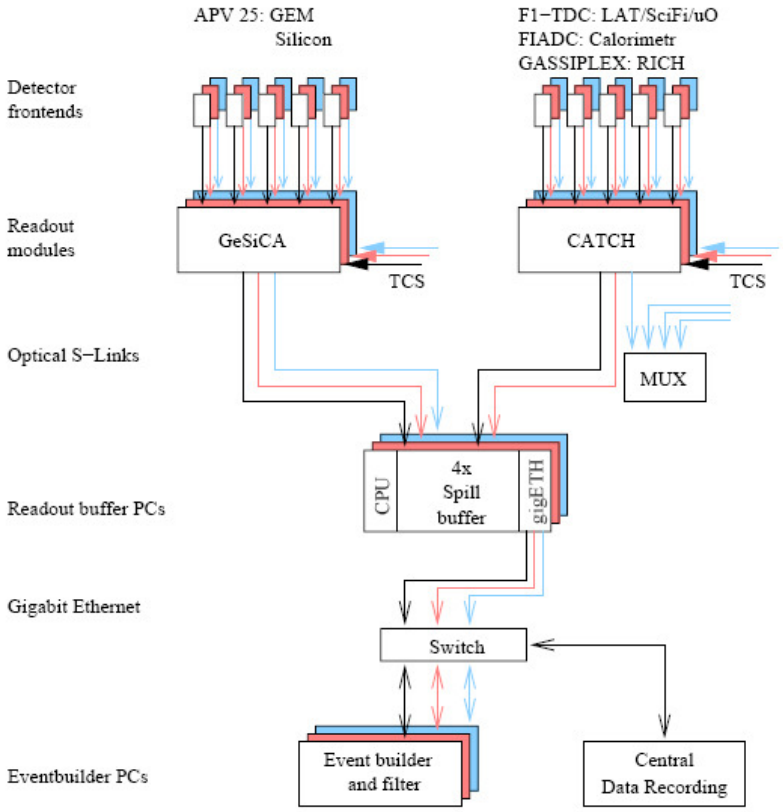
V současné době je připraven koncept návrhu na pokračování tohoto experimentu po dalších 5 let se třemi základními programy měření.

PROCES SBĚRU DAT V EXPERIMENTU

Svazek urychlených částic se skládá z periodicky se opakujících úseků, tzv. spillů. V průběhu jednoho spillu projde tok obsahující přibližné 2×10^8 částic. Celkový objem dat získaných při průletu jedné částice zařízením je cca 35 kB. Data získaná sledováním vyletujících částic přichází od detektoru v 250 000 kanálech. Množství dat tak dosahuje zhruba 580 TB za rok.

Na obrázku 1 je znázorněno schéma toku dat od elektroniky detektoru k permanentnímu úložišti. Součástí detektoru je primární elektronika starající se o produkci dat. Data od detektorů přicházejí se zpožděním v závislosti na vzdálenosti od terče.

Systém zpracování dat se skládá z několika vrstev počítačů.



Obrázek 1: Jednotlivé vrstvy systému pro sběr dat

Primární elektronika detektorů, jež tvoří první vrstvu, tato data synchronizuje, digitalizuje a posílá do druhé vrstvy tvořené počítači CATCH a GeSiCA, které je zpracují a spojí je s tzv. hlavičkou události. Ta je využita pro sestavení dat o částici z různých detektorů. Data se přenášejí přes optický systém S-Link, vyvinutý v CERNu, do třetí vrstvy, kterou tvoří tzv. Read-

Out Buffery (ROB). Přenosová rychlost dosahuje hodnot až 160 MB/s. ROB představují v podstatě vyrovnávací paměti. Data se nahrávají do paměti pouze během spillů a zpracovávají se efektivně i během přestávky mezi spilly.

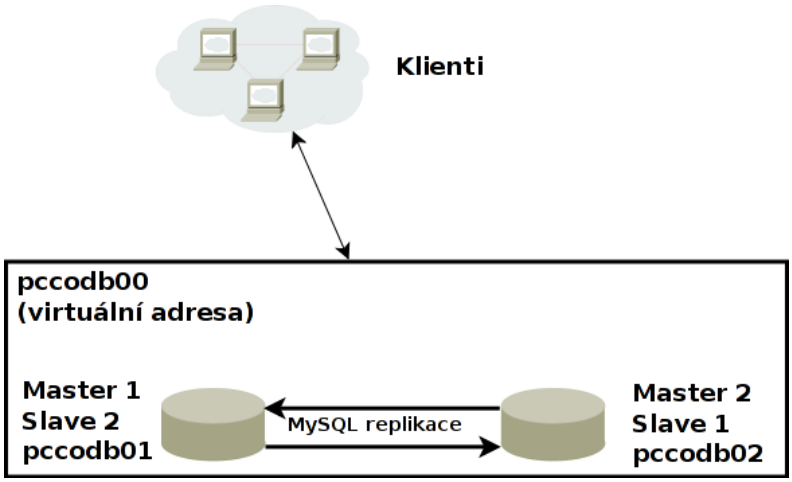
Poslední vrstvu tvoří tzv. event buildery (EVB), které spojují fragmenty dat z různých ROB do bloků popisujících průlety jednotlivých částic – tzv. událostí (event). Souběžně se načítají hlavičky událostí a ukládají se do metasouboru pro pozdější uložení do databáze. Po sestavení události se nashromážděná data nahrávají na lokální disky, načež jsou nástroji CDR (Central Data Recording) přenášena do počítačového centra CERN, kde se nachází CCF (COMPASS Computing Farm). Zde jsou uchována v permanentním úložišti nazývaném CASTOR (CERN Advanced STORage manager).

AKTUÁLNÍ DATABÁZOVÁ ARCHITEKTURA

V současné době jsou na COMPASSu využity dva fyzické databázové servery pccodb01 a pccodb02, které jsou vzájemně synchronizovány. Na obou je shodne instalován databázový server MySQL. Díky tomu se v případě výpadku serveru pccodb01 (na který je zapisováno implicitně) dá pro ukládání dat použít i „zálohovací“ server pccodb02.

Přístup z DAQ systému DATE je realizován přes virtuální adresu pccodb00, která přesměruje požadavek na pccodb01, pokud je v pořádku, a v případě přetížení ho přesměruje na pccodb02. Na serveru pccodb01 (resp. totožném pccodb02) je umístěno několik databází. Nejrozsáhlejší databáze DATE2009_logs obsahuje systémové informace z průběhu událostí v r. 2009, dále devdb je front-endová databáze s konfigurací elektroniky, databáze runlb – run log book – obsahuje zprávy ze směn experimentu a druhá nejrozsáhlejší beamdb (15 GB) ukládá detaily o každém ze spillů. Zápis tak probíhá v několikavteřinových intervalech vždy po ukončení spillu.

Naším úkolem je zejména vytvořit novou architekturu propojení serverů pro ukládání dat, která umožní rozdělení dvou největších databází (_logs a beamdb) na dva virtuální databázové stroje složené vždy ze dvojice strojů fyzických. Nyní jsou kopie obou databází shodně uloženy na obou strojích. Práce s rozsáhlými databázemi na fyzikálním experimentu však obsahuje mnohá specifika, která je třeba v návrhu uvažovat.



Obrázek 2: Stávající řešení

SPECIFICKÉ PROBLÉMY PŘI ZPRACOVÁNÍ DAT VE FYZICE VYSOKÝCH ENERGIÍ

Před návrhem nové databázové architektury bylo nezbytné otestovat možnosti serveru MySQL a klienta. Pro účely testu posloužila poměrně jednoduchá tabulka tbl_transact s následující strukturou:

```
CREATE TABLE tbl_transact (
  ID_rec bigint(20) AUTO_INCREMENT PRIMARY KEY NULL,
  NUM_CLI VARCHAR(255),
  COD_IP BIGINT(20),
  COD_CAT VARCHAR(255),
  QTY FLOAT,
  CAB FLOAT,
  CAN FLOAT,
  CAR FLOAT
);
```

Prostřednictvím skriptu v jazyce PHP bylo do tabulky vloženo až 25 milionů řádku vygenerovaných podle následujících pravidel: počet různých hodnot sloupce NUM_CLI je 30 000, počet různých hodnot ve sloupcích COD_IP a COD_CAT je 10 000; hodnoty ve sloupcích jsou čísla z intervalu [0, 200] uložená s přesností na 2 desetinná místa.

Na touto tabulkou byl spouštěn následující dotaz:

```
SELECT NUM_CLI, COD_IP, COD_CAT, sum(QTY) as S1,
sum(CAB) as S2, sum(CAN) as S3, sum(CAR) as S4 FROM
tbl_transact GROUP BY NUM_CLI, COD_IP, COD_CAT ORDER BY NULL;
```

Níže uvedená měření jsme prováděli na notebooku s dvoujádrovým procesorem Intel Core2 Duo T9600 běžícím na 2,8 GHz a s operační pamětí 4 GB. MySQL server ve verzi 5.1.44 běžel pod 64bitovým operačním systémem (Arch Linux, jádro 2.6.32).

V grafu na obr. 3 je vynesena závislost doby potřebné k vyhodnocení příkazu SELECT na počtu řádků v tabulce tbl_transact. Pro malé počty řádků doba zpracování škáluje zhruba lineárně, zlom nastává zhruba u 5 000 000 řádků. V tomto bodě dochází k vyčerpání operační paměti a začíná se uplatňovat řádově pomalejší odkládací prostor na disku. Pro tabulku s více než 10 000 000 řádky už nestačí ani odkládací prostor a proces MySQL klienta je mechanismem OOM (Out of memory) operačního systému ukončen.

Pomocí příkazu EXPLAIN lze snadno ověřit, že při pro vyhodnocení tohoto dotazu je procházena celá tabulka a jsou vytvářeny dočasné tabulky, což jsou časově i paměťově náročné operace. Dále jsou rozebrány techniky optimalizace, které mohou vést ke značnému urychlení dotazu.

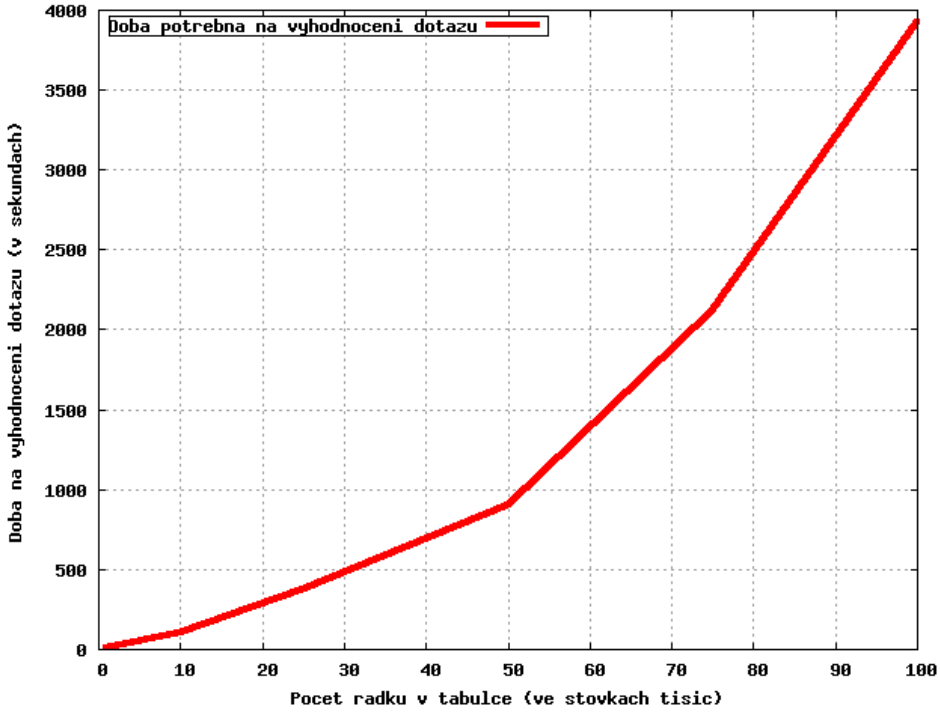
Nejprve jsme se pokusili použít různými způsoby index. Zatím jediný způsob, kterým se nám podařilo index aplikovat, abychom nemuseli používat dočasné tabulky, byl případ, kdy jsme do indexu vložili všechny sloupce dotazu, tedy bohužel všechny sloupce tabulky. Celý index ale nemůže přesahovat 1000 bajtů, proto jsme museli sloupce s 255znakovými řetězci zkrátit. Pro dlouhé řetězce jsme v MyISAM nenalezli vhodné použití indexu, zakomponování pouze prvních několika znaků do indexu se nejevilo jako efektivní. Se zkrácenými řetězci však dotaz proběhl až 5× rychleji na tabulce s indexem.

Ukázala se zde ale komplikace např. při příkazu INSERT (neúnosná doba vkládání nových dat), navíc významně vzrostla velikost celé databáze na disku. Při použití chybného indexu

(index je vytvořen na nevhodných sloupcích a dotaz se realizuje pomocí dočasných tabulek) je pak celý příkaz SELECT ještě pomalejší než původně, v lepším případě stejný.

Použití dělení tabulky (partition) také nepřineslo významné urychlení, což je pravděpodobně způsobeno tím, že test probíhal na systému s jediným diskem. Hash partitioning na 300 tabulek dle primárního klíče (ID_rec) nemělo na rychlost průběhu dotazu vůbec žádný vliv.

Dále jsme testovali změnu datového typu varchar(255) na pevný char(255), který by měl podle [5] mít menší režii. Nicméně vzhledem k tomu, že délky řetězců byly generovány náhodně od 0 do 255 a tedy v průměru byly poměrně krátké, se po použití typu char velikost databáze 5× zvětšila a celý dotaz tak probíhal pomaleji než dříve (pro 2 500 000 řádků původní dotaz probíhal 6 min, nový 9 min). Úprava datového typu float na decimal se žádnou výraznější úsporou neprojevila.



Obrázek 3: Závislost doby zpracování dotazu na velikosti tabulky

Na druhé straně zkrácení maximální délky řetězců ve sloupcích NUM_CLI a COD_CAT z 255 na 25 přineslo významné úspory v diskovém prostoru zabraném tabulkou a také v době potřebné na vyhodnocení dotazu. Snížení velikosti ukládaných dat doporučuje jako jeden z nejvýznamnějších faktorů při optimalizaci i oficiální dokumentace k MySQL [5]. Zkrácení sloupců u tabulky s 5 miliony řádků vedlo ke snížení použitého diskového prostoru z 1,5 GB na necelých 400 MB, doba potřebná k vyhodnocení dotazu ze zkrátila z 935 s na 385 s, tedy zhruba 2,5×. Navíc lze nad takto upravenou tabulkou vytvořit index a dotaz tak ještě zrychlit. Zkrácení maximální délky sloupců ovšem nemusí být vždy možné.

Dále se testovala změna úložiště z původního MyISAM na InnoDB. To umožnilo vytvořit nad původní tabulkou index ze všech sloupců, protože InnoDB nemá na rozdíl od MyISAM omezení 1000 bajtů na index. Přestože bez použití indexu proběhl celý dotaz na InnoDB pomaleji než na MyISAM, výše zmíněná vlastnost po použití indexu umožnila realizovat výběr dat z původní 20 GB tabulky v přijatelném čase, aniž by byl klient ukončen („zabit“). Systém InnoDB používá transakční zpracování, je tedy podstatně pomalejší při operacích jako vkládání, úpravy řádků nebo vytváření indexu.

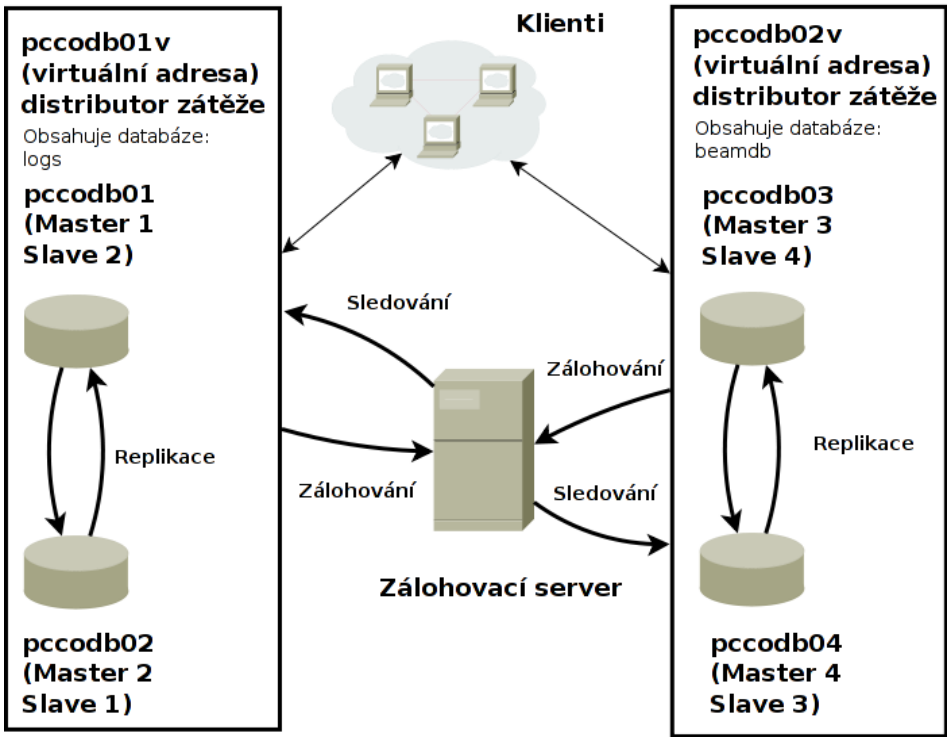
Operace/Úložiště	MyISAM	InnoDB
Naplnění tabulky	373 s	1945 s
SELECT (tab. bez indexu)	906 s	2042 s
Vytvořenii indexu	N/A	10740 s
SELECT (tabulka s indexem)	N/A	25 s

Tabulka 1: Srovnání úložiště MyISAM a InnoDB

Výsledky srovnání výkonu obou úložišť na tabulce s 5 miliony řádků jsou uvedeny v tabulce 1. Úložiště InnoDB bychom použili v případě, že by se tabulka naplnila jednorázově a později už by se z ní data pouze dolovala příkazem SELECT nebo v případech, kdy je požadováno transakční zpracování. V případě častého zápisu je vhodnější zvolit úložiště MyISAM.

NAVRHOVANÁ NOVÁ ARCHITEKTURA

Návrh nového řešení je znázorněn na obrázku 4. Vzhledem k velikosti dvou největších databází (logs, beamdb) navrhujeme jejich rozdělení na dva oddělené virtuální databázové stroje pccodb01v a pccodb02v. Každý z těchto virtuálních serverů by se skládal ze dvou fyzických strojů vzájemně synchronizovaných metodou MySQL replikace. Jeden pár serverů by obsluhoval databázi logs, druhý databázi beamdb. Ostatní menší databáze bychom rovnoměrně rozdělili mezi servery. Návrh také počítá s vyvažováním zátěže prostřednictvím programu MySQL Load Balancer a také se samostatným zálohovacím serverem, který by zároveň sloužil ke sledování celého systému.



Obrázek 4: Návrh nové architektury

Podle obrázku 4 se virtuální stroj pccodb01v skládá z fyzických strojů pccodb01 a pccodb02. Server pccodb02 slouží jako master, pccodb02 jako slave. Současně také pccodb01 funguje jako slave master serveru pccodb02. Toto uspořádání známé jako master-master replikace [7] zajišťuje vysokou dostupnost.

Pro rozdělení zátěže mezi fyzické servery plánujeme nasadit jako distributor zátěže aplikaci MySQL Load Balancer [6]. Tato aplikace se skládá ze dvou součástí označovaných jako prostředník a hlídač. Hlídač sleduje dostupnost a zatížení zúčastněných MySQL serverů a předává informace prostředníkovi, který podle těchto informací připojuje klienty k méně vytíženým serverům. Load Balancer používá stejný komunikační protokol jako MySQL server, funguje tedy jako proxy server mezi MySQL servery a klienty.

Load balancer prozatím umí rozdělovat dotazy, které nemění data (tedy SELECT). Load balancer se spouští příkazem mysql-lb, kterému se předá adresa replikačního master serveru a adresy slave serverů; v tomto návrhu:

```
mysql-lb --proxy-backend-addresses=pccodb01
         --proxy-read-only-backend-adressess=pccodb02
         --proxy-lua-script=monitored-ro-balance.lua
         --monitor-lua-script=monitor-backends.lua
```

Veškeré dotazy, které mění data (INSERT, UPDATE, DELETE), tedy budou automaticky odeslány na server pccodb01, dotazy SELECT distributor zátěže rozdělí mezi servery podle jejich zatížení. Třetí a čtvrtý parametr aplikace mysql-lb představují skripty v jazyce Lua, které plní funkci prostředníka, respektive hlídače. Úpravou těchto skriptů lze přeprogramovat chování aplikace.

V případě pádu serveru pccodb02 bude distributor zátěže předávat všechny dotazy na server pccodb01 až do doby, kdy pccodb02 nastartuje a synchronizuje se. Pokud ovšem selže server pccodb01, bude muset dojít ke změně nastavení distributoru zátěže tak, aby dotaz směroval na pccodb02. K tomu postačí ve výše uvedené ukázce zaměnit hodnoty parametrů proxy-backend-addresses a proxy-read-only-backend-adresseses. Je tedy zřejmé, že se musí zajistit neustálý dohled nad systémem.

Stejné uspořádání platí i pro virtuální server pccodb02v skládající se ze serverů pccodb03 a pccodb04.

Na databázových serverech nastavíme démon cron tak, aby každých 24 hodin spustil program mysql-dump pro vytvoření kompletní zálohy všech databází. Tyto zálohy budou pomocí nástroje scp překopírovány na oddělený zálohovací server – nejprve ze stroje pccodb01v a následně ze stroje pccodb02v. Vždy budou ponechány alespoň dvě poslední zálohy.

Na zálohovací server se dále nainstaluje monitorovací software Nagios. Tímto systémem se bude sledovat přinejmenším stav databázových serverů, volné místo na discích a stav démonu cron. V případně detekce problému se provede nějaká akce (například se upraví nastavení distributoru) a pošle se textová zpráva a e-mail zodpovědným osobám. Není třeba sledovat pokusy o neoprávněné průniky do systému, protože zabezpečení je řešeno na úrovni celé sítě v CERN.

WWW stránky experimentu COMPASS také používají několik menších databází (například seznamy schůzek jednotlivých skupin). Navrhujeme tedy přesunout tyto databáze na jeden z virtuálních serverů pccodb01v, pccodb02v.

Nakonec uvádíme stručný soupis činností, které je potřeba při realizaci návrhu provést:

1. prodiskutování a schválení návrhu
2. získání nových serverů
3. instalace a konfigurace těchto strojů
4. testování nastavení
5. export dat na starých, záloha nastavení
6. přesun a import dat na nové servery

7. podrobná kontrola data (přesná shoda)
8. údržba

ZÁVĚR

Databáze používané v experimentech z oblasti fyziky vysokých energií se od běžných komerčních databází liší svým rozsahem, strukturou i frekvencí nutných přístupů k datům. Jejich specifika je proto nutné při tvorbě návrhu neopomíjet a zvažovat možný dopad jednotlivých obvykle používaných řešení na ně. Tomu je třeba návrh primárně přizpůsobovat a standardní řešení modifikovat pro použití na takto rozsáhlých a vysoce využívaných databázích.

Již při návrhu schématu je třeba myslet na minimalizaci požadavků na velikost uložených dat, tj. využívat co nejúspornějších datových typů. Při analýze samotných dotazů je pak vhodné používat příkazů EXPLAIN (EXTENDED) a ANALYZE TABLE, ze kterých lze získat dostatek potřebných informací pro naplánování, jakým způsobem zamezit neúspornému prohledávání celé tabulky a kdy je např. možné na dané tabulce použít indexovou metodu. Vytvoření velkého indexu je sice paměťově náročné a zpomaluje vkládání nových dat, ale při vyhledávání dat již uložených práci výrazně urychluje.

Analyzovali jsme současnou architekturu databázových systémů používaných při sběru dat na experimentu COMPASS a připravili návrh na vylepšení této architektury. Tento návrh bude začátkem jara 2010 prodiskutován v rámci kolaborace COMPASS a následně realizován, pokud možno ještě před začátkem letošního sběru dat.

PODĚKOVÁNÍ

Práce na tomto příspěvku byla podporována z grantu MŠMT LA08015 a SGS 10/094.

LITERATURA

[1] Gurzhiy, E.: *Řízení distribuovaných výpočtů při zpracování experimentálních dat*. Bakalářská práce. Praha: ČVUT, 2007.

[2] Liška, T. *The Cluster and GRID Computing on Mass Data Production*

Systems for High Energy Physics Experiments. Doktorská disertační práce. Praha: ČVUT, 2009.

[3] Král, A, Liška, T, Virius, M.: *Experiment COMPASS a počítače*. In: Československý časopis pro fyziku 2005, č. 5, str. 472.

[4] *COMPASS page* [online]. 2010 [cit. 25. 2. 2010].

Dostupné na: `http://wwwcompass.cern.ch`

[5] *MySQL 5.1 Reference Manual* [online]. 2010 [cit. 19. února 2010].

Dostupné na: `http://dev.mysql.com/doc/refman/5.1/en/`

[6] *MySQL Load Balancer Guide* [online]. 2010 [cit 8. března. 2010]. Dostupné na: `http://downloads.mysql.com/docs/mysql-load-balancer-en.a4.pdf`

[7] *Linux Tutorials - MySQL Master_Master Replication* [online]. 2010 [cit.

19. února 2010]. Dostupné na:

http://www.howtoforge.com/mysql_master_master_replication