Hadoop Virtualizat

Dragoș-Bogdan Sima

Abstract

Apache Hadoop este un Framework Software pentru cloud computing, ce permite scrierea și executarea aplicațiilor distribuite ce procesează cantități mari de date. Aplicațiile se pot executa într-un mediu format din sute sau chiar mii de sisteme de calcul independente ce conlucrează la rezolvarea unor probleme (job-uri). Hadoop a fost conceput să rezolve problema defecțiunilor hardware într-un mediu distribuit. Aftfel, nefuncționarea unei mașini fizice sau a unui rack de mașini din cadrul sistemului distribuit este tratată în software de către framework evitând esecul execuției aplicației în curs. Fiind un proiect open source, licențiat sub Apache Licence 2.0, Hadoop este folosit și imbunătățit de o comunitate mare de dezolvatori și utilizatori. Lucrarea prezintă o analiză a performanțelor rulării unei aplicații în Hadoop, atât într-un cadru general (nativ), cât și într-un mediu specific (virtualizat), urmărind comportamentul pe exemple reale. Pe langă faptul că este cu sursă deschisă, framework-ul se remarcă prin usurința de a configura mediul de dezvoltare și de a specifica aplicația ce se dorește a fi rulată. Sunt evidente totodată performanțele bune atinse prin utilizarea unui mediu distribuit de calcul. Pe viitor se dorește ca munca realizată de catre comunitate să fie confirmată în continuare prin dezvoltarea unor componente care să crească atât nivelul tehnologic cât și cel informațional, și de ce nu, să contribuie esențial în cercetarea din orice domeniu unde este nevoie de prelucrarea unui volum foarte mare de date. Cuvinte cheie: Hadoop, Map-Reduce, HDFS.

1 Introducere

În prezent progresele în domeniul tehnologiei electronice se află într-o continuă creștere impunand și oferind totodată posibilitatea produselor software să țina pasul. De asemenea, dezvoltarea rapidă a fenomenului the web a adus cu sine o avalanșa de informații. Pentru un numar mare de organizații, aplicațiile pe care le dezvoltă sau folosesc au la bază prelucrarea unui set de date care, precum spuneam, este în permanența creștere. Aceste tip de aplicații ce necesită o prelucrare intensivă a datelor - data intensive problems - au nevoie de high-throughput I/O (transfer mare de date de intrare și ieșire) pentru a putea fi maleabile. Hadoop a fost conceput să ruleze aplicații data-intensive pe arhitecturi de tip cluster, și să ofere o mult mai bună scalabilitate datorită sistemului său distribuit de fișiere.

În condițiile în care domeniul electronic evoluează rapid, pentru rularea unor aplicații data-intensive, se pune problema dacă se merită folosirea unor componete harware puternice, dar scumpe, sau crearea unui cluster din componente la un preț rezonabil. În marea majoritate a cazurilor, cea de-a doua variantă va fi cea ideală, iar un framework pentru calcul distribuit precum Hadoop, a devenit unul dintre cele mai populare instrumente pentru a administra și transforma cantitati mari de date într-o maniera care să rezolve problema costurilor. Astfel, paradigma presupune ca fiecare disk, server, network link, sau chiar rack din cadrul cluster-ului poate să nu funcționeze la un moment dat.

Construcția și capabilitățile cu care vine Hadoop - de a manipula eficient cantități mari de date l-au transformat într-o platformă de integrare, transformare și de analiză a informațiilor. Astfel, Hadoop poate fi utilizat pentru o varietate de tipuri de aplicații și necesități:

- **Personalizarea conținutului pentru utilizatori**: Crearea unei experiențe mai bune prin reclame relevante, personalizarea home page-urilor sau recomandări bune.
- **Managementul unui lanț de aprovizionare** Examinarea datelor istorice disponibile permite decizii mai bune în privința stocurilor și administrarea bunurilor; printre sectoarele ce au nevoie de astfel de date se numară comerțul, agricultura, cel de sanătate sau energetic.
- **Analiza fraudelor**: Cercetarea tranzactiilor istorice din sectorul financiar (de exemplu, carduri bancare sau ATM-uri) pentru a detecta inselaciunile;
- **Bioinformatică**: Analiza genomului sau aplicarea unor algoritmi de secvențializare ADN asupra unor baze de date mari.
- **Diferite alte utilizări**: Agregarea unor seturi mari de imagini obținute din diferite surse și combinarea acestora într-una singură (de exemplu, imaginile din satelit), mutarea unor cantități mari de date dintr-o locație în alta, ș.a.

Pentru a putea analiza performanțele unei tehnologii, trebuie mai întâi înțeles modul de funcționare al sistemului, și daca este cazul, al componentelor. Pentru Hadoop, în particular, este esențială înțelegerea a două elemente: cum sunt depozitate și cum sunt procesate datele. Hadoop este compus din 2 concepte

de bază: un sistem de fișiere distribuit (inspirat după **Google File System**) și un framework computațional (**Google Map-Reduce**). Datele sunt ținute în prima componentă - Hadoop Distributed File System (HDFS). În versiunea Apache open-source de HDFS, namespace-ul DFS este administrat de un singur NameNode, în timp ce un set de DataNodes are rolul de a stoca și returna blocuri de date în rețea. HDFS administrează deasemenea replicarea blocurilor de data, pentru a evita pierderea datelor în cazul nefuncționării unui DataNode. Însă, deși astfel Hadoop asigură protecția datelor, NameNode-ul este singura verigă care poate conduce către nefuncționarea corectă a framework-ului. De aceea se recomandă ca acest nod să fie izolat și să se folosească diferite măsuri pentru a asigura disponibilitatea acestuia. Există un NameNode secundar care conține o copie a datelor de pe NameNode-ul principal, ce vor fi folosite pentru a-l reporni în caz de nefuncționare, însă este posibil ca copia să nu conțina ultima variantă a datelor, și deci să se piardă unele informații. Sistemul de fișiere folosește nivelul TCP/IP pentru transferul blocurilor, și RPC pentru comunicare între DataNode-uri.

Cea de-a doua caracteristică a proiectului Hadoop este abilitatea de a procesa datele stocate în HDFS, sau cel puțin de a oferi un framework pentru îndeplinirea acestui țel. Aceasta este MapReduce care, în loc să folosească modul convențional de a muta datele în rețea - proces ce poate fi foarte încet, în mod special pentru date mari - către locul unde este software-ul de procesare, abordează o altă perspectivă, aceea de a muta algoritmul către date. În această paradigma de procesare computațională a datelor, un singur JobTracker programează job-urile din cluster, precum și task-urile individuale. Un job este format dintr-un set de task-uri ce sunt executate de catre nodurile worker. Pe fiecare nod rulează un TaskTracker care este responsabil pentru a porni task-urile și de a raporta progresul către JobTraker. Așa cum sugereaza și numele, procesarea MapReduce este compusă din două faze: map și reduce. Ambele folosesc perechi cheie-valoare, definite de către utilizator, ca parametri de intrare și ieșire. Această convenție permite ca rezultatul unui job să fie folosit ca intrare pentru un alt job, operație des folosită în Hadoop. Numărul de task-uri de tip map și reduce sunt independente, și nu este obligatoriu ca un set de task-uri să folosească toate nodurile din cluster. În Figura 1, este este exemplificat dataflow-ul cu doar un task per nod.

Hadoop împarte datele de procesat în chunk-uri pentru procesarea de catre task-uri de tip map. Aceste împărțiri trebuie să fie suficient de mari încât să minimizeze overhead-ul de a administra fișiere și task-uri, dar îndeajuns de mici pentru a exista o paralelizare a muncii pe noduri. Rezultatul task-ului de tip map este sortat dupa cheie și nu este scris în HDFS, ci local pe disk (cu gri in Figura 1). Apoi este trimis către task-ul reduce corespunzător. Această etapă este numită "shuffle phase", tipică operațiilor de tip all-to-all care de multe ori forțeaza bandwidth-ul (lațimea de bandă) rețelei ce interconectează sitemele. Task-urile de tip reduce fac ultima procesare, rezultatul fiind scris în HDFS (cu albastru în Figura 1). În cazul în care există mai mult decât un reducer, rezultatul este partiționat. O carte ce oferă mai multe detalii despre HDFS, Map-Reduce și alte componente, este Hadoop: **The Definitive Guide, 3rd Edition**.

Devremece joburile Hadoop pot fi foarte mari, unele având nevoie de ore să fie executate, chiar și pe clustere mari, costurile resurselor pot fi semnificative. Costului unui job este invers proporțional cu viteza de transfer pe care Hadoop

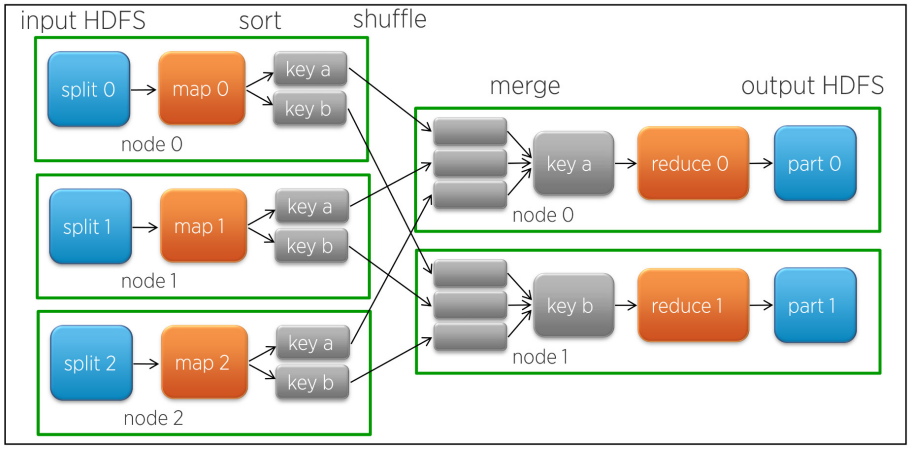

Figure 1: Simplified MapReduce Dataflow with One Task Per Node.

poate să o ofere, și prin urmare performanța este extrem de importantă. Procesorul, memoria, viteza de transfer în retea, și capacitatea și viteza de transfer a dispozitivelor de stocare trebuie să fie calibrate corespunzator pentru a obține un sistem balansat. De asemenea este necesar ca partea software să fie capabilă să utilizeze toate aceste resurse. În Hadoop acest lucru presupune paralelizarea muncii într-un mod care să țină nodurile computaționale ocupate, și programarea task-urilor de rulat într-o manieră care să presupună pe cât posibil ca datele necesare acestora să fie locale (mutarea calculului către date, și nu mutarea datelor către resursele de calcul). Overhead-ul sau ineficiența planificării în virtualizare, sistem de operare sau nivelul aplicației se pot contribui cu un cost cumulativ semnificativ, și prin urmare trebuie înțelese și cuntificate.

2 Virtualizare

Hadoop este o aplicație modernă cu caracteristici precum consolidarea job-urilor și high-availability, ce se suprapun cu capabilitățile oferite de către virtualizare. Deși astfel, se poate concluziona că virtualizarea Hadoop-ului nu este o soluția viabilă, totuși există o varietate de motive pentru această alegere. Unele dintre acestea sunt:

- **Planificarea**: Joburile de tip batch pot fi rulate în perioade în care capacitatea infrastructurilor virtuale este mai puțin folosită (de exemplu, noaptea).
- **Utilizarea resurselor**: Mașinile virtuale (VM) Hadoop și alte VM pot fi localizate pe aceleași sisteme gazdă (hosts). Acest lucru permite o utilizare generală mai bună, consolidând aplicații care folosesc tipuri diferite de resurse.
- **Modele de stocare**: Deși Hadoop a fost gândit urmărind stocarea locală, acesta poate folosi cu usurință memorie comună (shared memory) pentru toate datele sau un model hibrid în care datele temporare să fie ținute

local, pe disk iar HDFS să fie localizat pe un SAN (Storage Area Network). Alegând oricare dintre cele două configurări, capacitatea și lațimea de bandă a memoriei comune, nefolosită din cadrul infrastructurii virtuale poate fi folosită de joburile Hadoop.

- **Eficiența unui datacenter**: Virtualizând Hadoop se poate îmbunătați eficiența datacenter-ului prin creșterea numărului de tipuri de volum de muncă (workload) ce pot rula pe un sistem virtual.
- **Performanța**: Virtualizarea permite în esența cresterea flexibilității configurării resurselor hardware.

Acest ultim atribut este și principalul punct tratat în această lucrare. Multe aplicații pot să folosească eficient un numar mare de noduri într-un cluster, scalând, dar nu reușesc cu succes să îmbunătățească scalarea SMP (Symmetric multiprocessing). Mai exact, nu scalează vertical - adăugând resurse unui singur nod. O varietate de servere web, de mail sau aplicații Java și nu numai intră în această categorie. Într-un mediu virtual, această problema se poate rezolva ân general prin rularea mai multor mașini virtuale mici per host.

Oricare ar fi motivul ales pentru virtualizarea Hadoop-ului, este importantă înțelegerea performanțelor implicate de o astfel de decizie pentru a asigura costuri rezonabile și resurse suficiente. Deși cele mai relevante probleme sunt investigate în acest articol, rămân multe altele de tratat în rapoarte ce vor urma. Printre acestea se numară compromisuri asupra diferitelor tipuri de stocare, adaptarea rețelei, scalare pe orizontală (adaugare de noduri) mai mare, și mai multe tipuri de aplicații.

3 Configurarea sistemului de test

Configurarea hardware a cluster-ului de testare este aratată în Figura 2. Cele 7 servere gazda dintr-un system AMAX ClusterMax au fost conectate la un singur switch Mellanox 10GbE. Fiecare gazdă a fost echipată cu 2 procesoare Intel X5650 2.66GHz 6-core, 96GB RAM, 12 disk-uri SATA 500GB 7,200 RPM, și câte un adaptor 10GbE Mellanox. Disk-urile au fost conectate la un storage controller PCIe 1.1 X4. Acest controler are, teoretic, o viteză de transfer de 1GB/s, dar practic poate să suporte 500-600MB/s. Starile de putere (power states) au fost scoase din BIOS pentru a îmbunătăți consistența rezultatelor. Opțiunea Intel Hyper-Threading (HT) a fost invalidată în cazul unor teste. În privința sistemului de operare, s-a folosit RHEL 6.1 pe 64 biți pentru toate testele, fiind instalată versiunea Cloudera CDH3u0 de Apache Hadoop pe toate mașinile.

Cum scopul acestui document este analiza performanțelor Hadoop într-un mediu virtualizat, detalii despre configurarea hardware, sistemului de operare și a parametrilor Hadoop se găsesc în Anexa. Informații complete despre cum se poate reproduce configurarea, pas cu pas, pot fi oferite la cerere. De reținut este faptul că detaliile de configurare a sistemului au fost gândite în așa fel încât rezultatele benchmark-ului să fie atât relevante, cât și să ofere performanțe Hadoop cât mai bune.

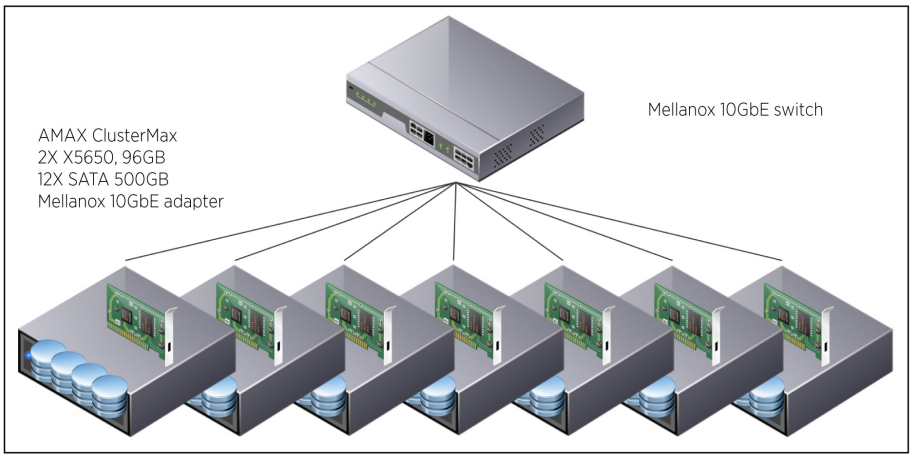

Figure 2: Cluster Hardware Configuration.

BENCHMARK	MAP		REDUCE	
	HT on	HT off	HT on	HT off
Pi	168	84	1	1
TestDFSIO-write	140	140	1	1
TestDFSIO-read	140	140	1	1
TeraGen	280	280	0	0
TeraSort	280	280	70	70
TeraValidate	70	70	1	1

Figure 3: Maximum Total Number of Simultaneous Tasks per Job.

4 Cazuri benchmark

Benchmark-urile au fost create din cele 3 exemple de aplicații din distribuția Cloudera: Pi, TestDFSIO și TeraSort. În Figura 3 se găsesc, pentru fiecare benchmark, numărul maxim de task-uri simultane de map și de reduce din cluster.

4.1 Pi

Pi este o aplicație pur computațională care folosește metoda Monte Carlo pentru a estima valoarea numărului pi. Este "jenant de paralelizabilă" (dacă mă pot exprima plastic): task-urile de tip map sunt toate independente și un singur task reduce colectează puținele date de la acestea. Așadar traficul în rețea este mic, iar operațiile de I/O sunt puține. Rezultatele raportate au reprezentat calculul a 1.68 de trilioane de sample-uri. Acestea au fost împărțite pe de o

parte, la 168 de mapper-i de cu HT, iar pe de alta, la 84 task-uri de tip map fara HT activat. Atât pentru cazurile omogemne, cât și pentru cele eterogene, numărul de task-uri de tip map per nod este egal cu numărul de CPU-uri sau vCPU-uri (Virtual CPU). Toți mapper-ii pornesc la același moment, iar job-ul este terminat în momentul în care ultimul task map își termină calculul, la care se adaugă un timp mic pentru reduce. Pentru o bună performanță este important ca task-urile de tip map să ruleze la aceeași viteză și să termine în timpi apropiați. Altfel, un task map lent poate avea un efect semnificativ asupra timpului de execuție al unui job.

4.2 TestDFSIO

TestDFSIO este un test pentru transferul de date, împărțit în două părți: TestDFSIO-write scrie aproximativ 1TB (1000020MB) de date în HDFS, și TestDFSIO-read care citește datele inapoi. Din cauza factorului de replicare, testul pentru scriere va avea de 2 ori mai multe operații I/O decât cel de citire, generând astfel un trafic în rețea destul de substanțial. În urma multiplelor teste pentru cazul fără HT validat, numărul optim de operații de tip map s-a dovedit a fi 140. Devremece puterea CPU-ului nu influențeaza semnificativ acest test dominat de operații I/O, același număr de task-uri fiind folosit și pentru folosirea HT. Numărul de mapper-i per nod în cazul eterogen a fost ales direct proporțional cu numărul de disk-uri disponibile (4 pentru VM mici, 6 pentru cele mari).

4.3 TeraSort

TeraSort este un algoritm de sortare a unui numar mare de înregistrari 100-byte. Astfel, pentru acest test va exista un volum mare de calcul, transfer în retea și operații I/O, fiind probabil cel mai relevant pentru un caz real al rularii unei aplicații în Hadoop. Acesta se împarte în 3 părți: generare, sortare si validare. TeraGen creaza datele, fiind similar cu TestDFSIO-write, cu precizarea că de data aceasta este nevoie de calcul semnificativ pentru crearea datelor random. Pentru că mapper-ii scriu direct în HDFS, nu este nevoie de faza de reduce. TeraSort sortează și scrie datele sortate în HDFS, în fișiere partiționate. Aplicația în sine suprascrie factorul de replicare specificat astfel încât doar o copie să fie scrisă. Filozofia este ca dacă un disk este nefolosibil, să se poată rerula aplicația, dar datele de intrare au nevoie de replicare devremece este posibil să nu fie atât de ușor de recuperat. TeraValidate citește datele sortate pentru a verifica că sunt ordonate. Mapper-ii fac această validare pentru fiecare fișier în parte, reducer-ul verificând că ultima înregistrare a fiecarui fișier este ordonat înaintea primei înregistrări din următorul fișier. Rezultatele expuse sunt pentru 10 miliarde de înregistrări (1TB), respectiv pentru 35 miliarde (3.5TB). Un număr total de 280 de task-uri de tip map au fost folosite pentru TeraGen, 280 mapper-i simultani și 70 de reduceri pentru TeraSort, și 70 de task-uri map pentru TeraValidate.

BENCHMARK	NATIVE	VIRTUAL	
		1 VM	2 VMs
Pi	792	740	762
TestDFSIO-write	640	706	614
TestDFSIO-read	499	532	453
TeraGen 1TB	664	700	580
TeraSort 1TB	2,995	3,127	3,110
TeraValidate 1TB	569	481	495
TeraGen 3.5TB	2,328	2,504	2,030
TeraSort 3.5TB	13,460	14,863	12,494
TeraValidate 3.5TB	2,783	2,745	2,552

Figure 4: Elapsed Time in Seconds with HT Disabled (lower is better; number of VMs shown is per host).

5 Rezultate benchmark

În Figurile 4 si 5 sunt prezentați timpii de execuție fară HT, respectiv cu HT activat.

Figurile 6 si 7 arată timpii rezultați pentru cazurile virtuale normalizate către cazurile native corespunzatoare, din nou fară HT, respectiv cu HT activat.

6 Analiză performanțe

6.1 Pi

Pi este, pentru cazurile virtuale, cu 4-10% mai rapid decât cele native corespunzătoare. Acesta este un rezultat neașteptat devremece în mod normal aplicațiile CPU intensive arată o reducere de 1-5% când sunt virtualizate. O contribuție la aceasta îmbunătățire a performanței provine din faptul că mapper-ii nu execută la aceeați rata (funcție a planificatorului Linux si ESXi) care este reflectată în cât timp este rulat reducer-ul (începe când primul task map a terminat). Diferențele de scheduler pot produce diferențe cu până la 1-2% asupra performanței. De asemenea, timpul total de execuție al mapper-ilor este mai mic cu 9% în cazul virtualizării. Având și o utilizare de 100% a procesoarelor, fiecare mapper rulează deci, mai repede într-o mașină virtuală. Folosind HT, timpul se reduce cu 12% pentru cazul nativ și 15-18% în cazul virtualizării.

6.2 TestDFSIO

TestDFSIO este un test de stress pentru aplicații I/O intensive. Rezultatele arată că ân 3/4 cazuri cu o mașina virtuală (write/read, HT disabled/enabled) timpul de execuție este cu 7-10% mai mic decât în cele native corespunzatoare. În cel de-al patrulea caz cu o VM și în toate celelalte cazuri cu multi-VM,

BENCHMARK	NATIVE	VIRTUAL		
		1 VM	2 VMs	4 VMs
Pi	695	628	626	650
TestDFSIO-write	648	715	573	562
TestDFSIO-read	471	443	433	427
TeraGen 1TB	659	718	572	598
TeraSort 1TB	2,629	3,001	2,792	2,450
TeraValidate 1TB	470	601	544	480
TeraGen 3.5TB	2,289	2,517	1,968	2,302
TeraSort 3.5TB	13,712	14,353	12,673	13,054
TeraValidate 3.5TB	2,460	2,563	2,324	3,629

Figure 5: Elapsed Time in Seconds with HT Enabled (lower is better; number of VMs shown is per host).

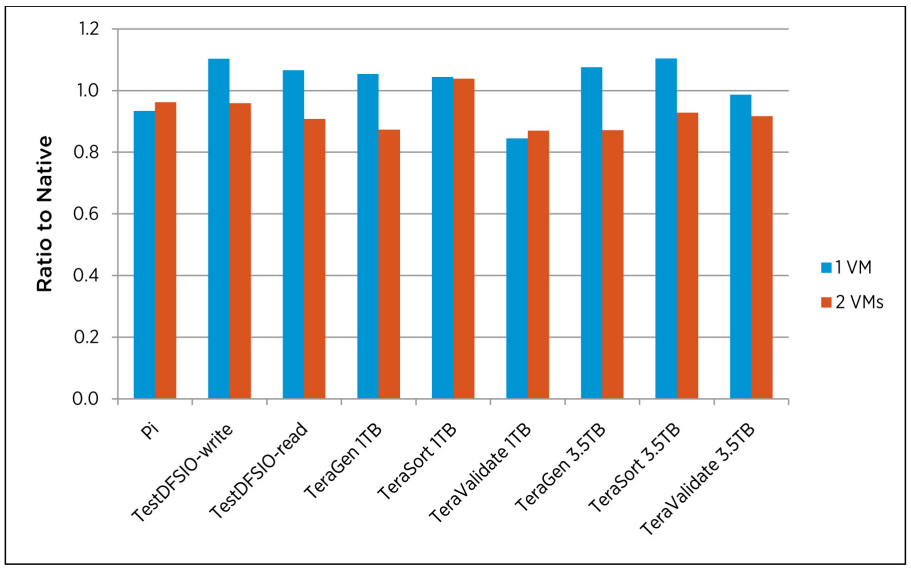

Figure 6: Elapsed Time with HT Disabled Normalized to the Native Case (lower is better).

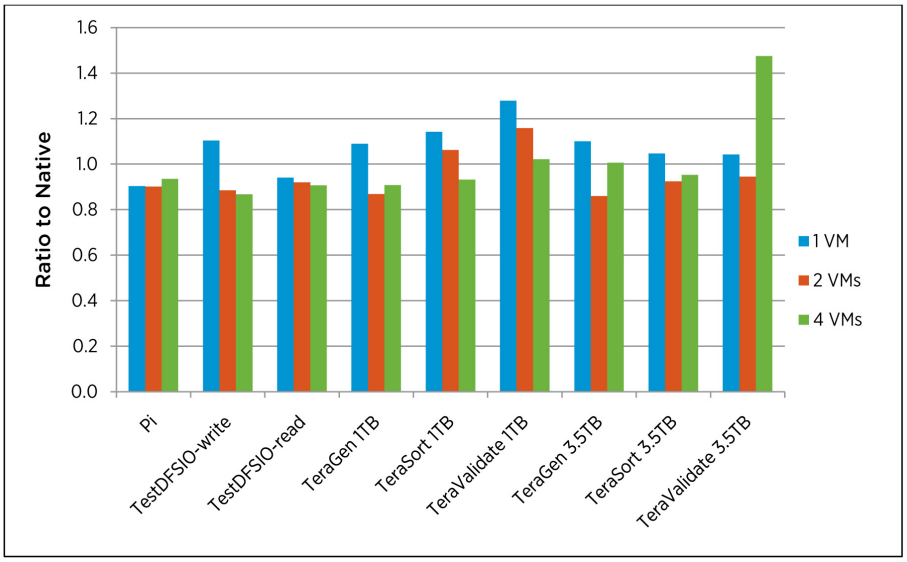

Figure 7: Elapsed Time with HT Enabled Normalized to the Native Case (lower is better).

virtualizarea are performanțe temporale cu 4-13% mai bune. Într-un astfel de test (transfer de date secvențial, cu o încărcare ușoară a procesoarelor) este de așteptat ca transferul de date să fie strict influențat de hardware, și ca varierea între platformele software să nu fie semnificativă. O explicație parțială a comportanemntului observat este evidențiat în Figura 8, care arată transferul la scriere pe una dintre gazde număr diferit de mașini virtuale și HT validat. Rata medie de transfer este mai scazută pentru cazul cu o mașina virtuală din cauză că transferul este mai zgomotos în timp, și nu pentru că vârful de transmitere este mai jos. De asemenea se observă că la anumite intervale transferul scade la 0, sincron pentru toate nodurile, ceea ce sugerează că sunt cauzate de aplicație și nu de platformă. Cu 2 mașini virtuale per host, troughput-ul este mult mai consistent, iar cu 4, și mai mult de atât. Cazul nativ prezintă zgomote asemenea celui cu o mașina virtuală. O posibilă explicație a îmbunătățirii ratei de transfer I/O pentru mai multe VM poate fi faptul că un singur nod Hadoop prezintă dificultați în a planifica fair pentru un anumit număr de disk-uri, și că se comportă mai bine cu mai puține disk-uri la dispoziție. Deși pentru rata de transfer a datelor este mult mai bună pentru 4 VM decât pentru 2, pentru o mare parte din durata de executare a job-ului, cazul cu 4 mașini virtuale oferă un timp de rulare cu doar 2% mai mic. Acest lucru se intamplă datorită eterogenității care împiedică ca disk-urile să se termine la momente apropiate, precum pentru 2 VM. Ajustând numărul de task-uri per nod, testul scrie un număr de fișiere unice pe fiecare nod proporțional cu numărul de disk-uri disponibile. Această proporționalitate este stricată însă de faptul că replicile sunt distribuite egal pe toate nodurile.

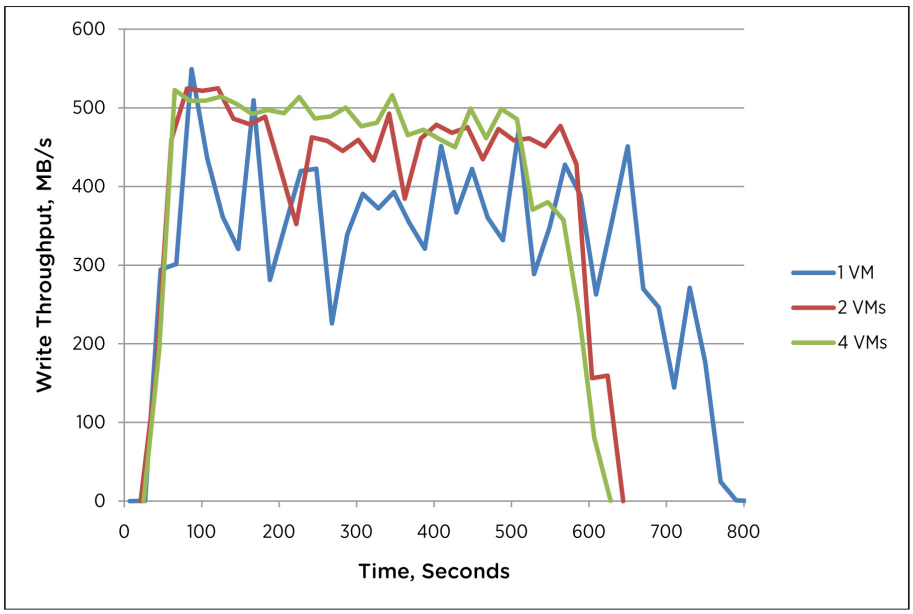

Figure 8: TestDFSIO Write Throughput on One of the Hosts (HT enabled).

6.3 TeraSort

Cele 3 benchmark-uri TeraSort arată buna performanță generală a virtualizării Hadoop.

Pentru a reduce zgomotele cauzate de complexitatea aplicației și de catre efectul "long pole" (un singur task ce durează mult influențează timpul total) testul cu un dataset de 3.5TB este cel mai potrivit.

TeraGen scrie aproape aceeași cantitate de date pe disk și are timpi similari pentru toate cazurile precum TestDFSIO-write. Nivelul de folosire a procesoarelor însa, este semnificativ mai mare (50% fata de 30%), dar nu este suficient de mare ca să afecteze timpul scurs.

TeraSort foloseste cel mai intens resursele: combină nivelul înalt de utilizare CPU, transfer mare de date și bandwith moderat în rețea (1-2 Gb/s per host). Folosind o singură mașină virtuală per sistem gazdă, virtualizarea are un timp cu 4-14% mai mic decât în cazul nativ. Reducerea de performanță cu 8% este rezonabilă pentru aplicații I/O intensive. Cu 2 VM, performanța este îmbunătățită semnificativ in comparație cu un caz nativ corespunzător: devine cu 4-6% mai slabă pentru 1TB și cu 7-8% pentru 3.5TB de înregistrări. HT îmbunătățește performanța cu 4-12% pentru 1TB, dar cu nimic semnificativ cazul cu 3.5TB. Cel de-al treilea caz cu 4 mașini virtuale oferă o creștere mare pentru cazul cu 1TB de date, însă destul de mică pentru 3.5TB de înregistrări.

Aceste observații pot fi explicate și prin utilizarea procesoarelor evidențiată în Figura 9 pentru cazul cu 3.5TB de date cu HT folosit. Mapper-ii și faza de shuffle ating preak-ul la aproximativ 8 000 secunde și după, reducer-ii rulează la un nivel de utilizare relativ mic. Efectul "long-pole" al puținelor task-uri de reduce, ce durează mult, cauzează în mare variația de timp scurs. Spre exemplu, cazul cu o mașină virtuală termină faza map/shuffle într-un timp apropiat de

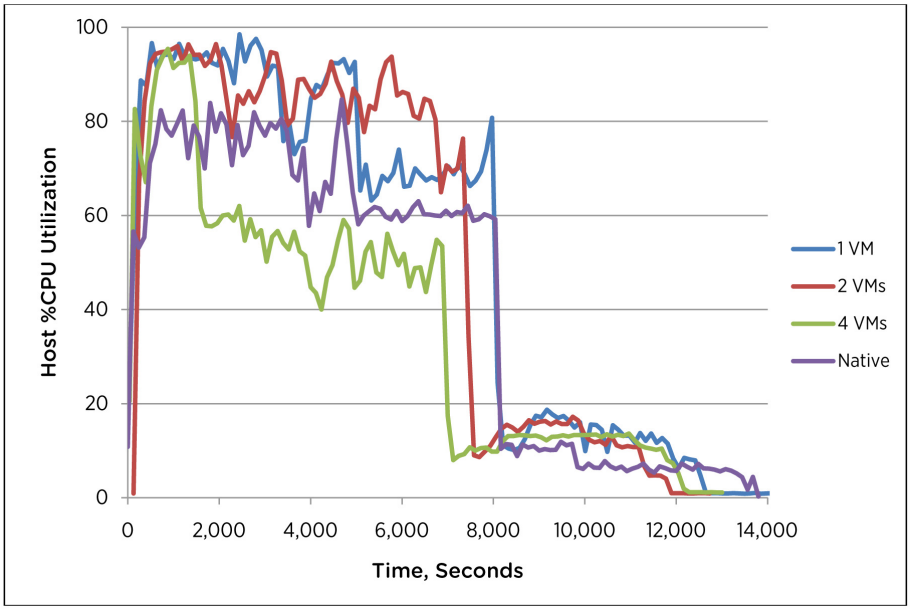

Figure 9: 3.5TB TeraSort %CPU Utilization with HT Enabled on One of the Hosts (lower is better).

cel nativ, și majoriatea task-urilor reduce mai rapid, însă întreaga rulare a job-ului durează mai mult. Cu 2 VM viteza de execuție se îmbunătățește pentru aproximativ aceeași utilizare a procesorului. 4 mașini virtuale cresc performanța map/shuffle, și pe deasupra, micșorează utilizarea CPU.

Un anumit tip de metrică de măsurare a performanței poate fi mai relevant decât oricare altă metrica pentru performanța aplicației. De exemplu, într-un mediu job-consolidation (rularea mai multor job-uri în Hadoop, sau alte tipuri de aplicații în alte mașini virtuale simultan), nivelul total de utilizare a procesorului este de obicei o metrică mai bună de performanță decât timpul scurs per job, devremece este necesară determinarea numărului de job-uri ce pot rula într-un anumit timp. Cu o astfel de metrică, avantajul performanței pentru 4 VM, față de o configurare nativa este mult mai relevant decât cele 5 procente indicate de către timpul de rulare al unui job. Cazul cu 1TB de date este similar în concluzii, deși metricile sunt diferite. Faza de map/shuffle folosește 96-99% din CPU pentru toate cazurile de virtualizare și aproximativ 75% pentru cel nativ. Câștigul folosirii mai multor mașini virtuale este reflectat în reducerea timpului total de execuție; cu 4VM faza de map/shuffle este cu 24% mai rapidă decat în cazul nativ pentru 1TB înregistrări, respectiv 8% pentru 3.5TB.

Una din caracteristicile Hadoop este execuția speculativă. Dacă un task rulează prea încet pe un nod și ale noduri sunt disponibile să preia un task, Hadoop va porni acel task pe un alt nod, păstrând rezultatul de la nodul care termină rularea cel mai rapid, și omorându-l pe celalalt. Strategia este eficientă dacă motivul pentru care un task rulează încet este o problemă la nivel de nod. Cazul extrem este când un nod este indisponibil, Hadoop repornind task-urile de pe acesta pe un alt nod. Însă dacă toate nodurile sunt funcționale, execuția speculativă poate încetini progresul job-ului. Acest lucru s-a întamplat pentru

toate cele 3 teste pe 4 mașini virtuale pentru testul cu 3.5TB de date. Eterogenitatea acestei configurări a condus la diferențe în executarea task-urilor, prin executarea speculativă. Acest efect este cel mai influent pentru TeraValidate: pornirea unui același task pe un alt nod este în general costisitoare deoarece există o singură copie a datelor ce trebuie să fie citite. Nefolosind executarea speculativă, TeraGen și TeraValidate obțin timpi de rulare îmbunătățiți cu 11%, respectiv 13% pentru 4VM și doar cu 5%, respectiv 7% pentru 2 mașini virtuale. TeraSort prezintă astfel, performanțe puțin îmbunătățite, probabil datorită numărului relativ mic de task-uri ce ajută la ocuparea permanentă a slot-urilor de task.

7 Concluzie

Aceste rezultate arată că virtualizarea Hadoop poate fi convingatoare chiar și numai din motive de performanță. Cu o singură mașina virtuală per sistem gazdă, creșterea mediei timpului scurs pentru toate testele de benchmark față de configurarea nativă este cu 4%. Acesta este un preț mic de plătit în comparație cu celelalte avantaje oferite de virtualizare. Rulând 2 sau 4 VM pe fiecare mașină fizică cu 2 socketi se obțin avantaje de performanța semnificativ mai bune fața de cazurile native corespunzătoare; în unele cazuri cu până la 14% mai bune. Folosind o metrică de utilizare a procesoarelor, ceea ce este de multe ori mai relevant pentru performanța unei aplicații, se pot genera câștiguri și mai mari.

8 Anexă

8.1 Hardware

- Cluster: AMAX ClusterMax with seven 2-socket hosts
- Host CPU and memory:
  - 2X Intel Xeon X5650 processors, 2.66 GHz, 12MB cache
  - 96 GB, 1333 MHz, 12 DIMMs
- Host storage controller:
  - Intel RAID Controller SROMBSASMR, PCIe 1.1 x4 host interface
  - 128 MB cache
  - Write Back enabled
  - Host hard disks:
  - 12X Western Digital WD5003ABYX, SATA, 500GB, 7200 RPM
  - 2 mirrored disks divided into LUNs for native OS and ESXi root drives and VM storage
  - 10 disks configured as JBODs for Hadoop data
- Host file system:
  - Single Linux partition aligned on 64KB boundary per hard disk
  - Partition formatted with EXT4:
    - 4KB block size
    - 4MB per inode
    - 2% reserved blocks
- Host network adapter: Mellanox ConnectX VPI (MT26418) - PCIe 2.0 5GT/s, 10GbE
- Host BIOS settings:
  - Intel Hyper-Threading Technology: enabled or disabled as required
  - C-states: disabled
  - Enhanced Intel SpeedStep Technology: disabled
- Network switch: Mellanox Vantage 6048, 48 ports, 10GbE

8.2 Linux

- Distribution: RHEL 6.1 x86_64
- Kernel parameters:
  - nofile=16 384
  - nproc=4096
- Java: Sun Java 1.6.0_25

8.3 Native OS

- CPU, memory, and disks: same as Hardware
- Network driver: Mellanox MLNX_EN (version 1.5.6), enable_sys_tune=1

8.4 Hypervisor

- vSphere 5.0 RTM, changeset 1401879
- Development version of Mellanox network driver

8.5 Virtual Machines

- VMware Tools: installed
- Virtual network adapter: vmxnet3
- Virtual SCSI controller: LSI Logic Parallel
- Disks: Physical Raw Device Mappings
- 1 VM per host:
  - 92000MB, 24 vCPUs (HT enabled) or 12 vCPUs (HT disabled), 10 disks.
- 2 VMs per host:
  - 46000MB, 12 vCPUs (HT enabled) or 6 vCPUs (HT disabled), 5 disks
  - One VM pinned to each socket.
- 4 VMs per host:
  - HT enabled only
  - Small: 18400MB, 5 vCPUs, 2 disks
  - Large: 27600MB, 7 vCPUs, 3 disks
  - One small and one large VM pinned to each socket

8.6 Hadoop

- Distribution: Cloudera CDH3u0 (Hadoop 0.20.2)
- NameNode, JobTracker: node0
- Secondary NameNode: node1
- Workers: All machines/VMs
- Non-default parameters:
  - Number of tasks: Table 1
  - dfs.datanode.max.xcievers=4096

  - dfs.replication=2
  - dfs.block.size=134217728
  - io.file.buffer.size=131072
  - mapred.child.java.opts="-Xmx2048m –Xmn512m" (native)
  - mapred.child.java.opts="-Xmx1900m –Xmn512m" (virtual)

- Cluster topology:
  - Native, 1 VM per host: all nodes in the default rack
  - 2, 4 VMs per host: each host is a unique rack

9 Referințe

1. Tom White, The Definitive Guide, 3rd Edition - http://www.amazon.com/Hadoop-Definitive-Guide-Tom-White/dp/1449311520

2. A Benchmarking Case Study of Virtualized Hadoop Performance on VMware 5 - http://www.vmware.com/files/pdf/VMW-Hadoop-Performance-vSphere5.pdf