BigDL - Distribuirano okruženje dubokog učenja za velike skupove podataka

David Stanojević

15. januar 2022.

Sažetak

Ovaj rad predstavlja BigDL [4] okruženje za razvoj aplikacija dubokog učenja u okviru Apache Spark [7] klastera. Sam BigDL je rezultat popularizacije dubokog učenja u oblasti vještačke inteligencije i značaja obrade velikih količina podataka u modernim cjevovodima dubokog učenja. Prvenstveno omogućava aplikacijama dubokog učenja da se pokreću na Apache Spark klasterima. Za razliku od postojećih okruženja za razvoj aplikacija dubokog učenja, BigDL implementira distribuirano i paralelno učenje modela zasnovano na postojećem Apache Spark modelu izračunavanja. U ovom radu pokušaće se dati odgovor na pitanje problema distribuiranog učenja modela u kontekstu velikih količina podataka i kako BigDL rješava taj problem.

1 Uvod

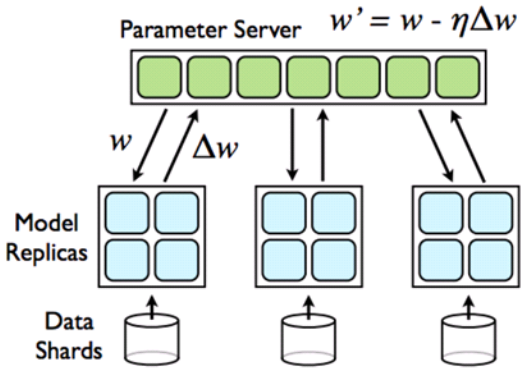
Kao rezultat značajnih napredaka u oblasti vještačke inteligencije i dubokog učenja kao glavnog alata za analizu kompleksnih podataka, pojavila su se mnogobrojna okruženja (npr. Torch, Caffe [1], TensorFlow [3], PyTorch [2]), koja se danas intenzivno koriste za razvoj modela dubokog učenja. Uprkos tome, postoje mnogobrojni izazovi za pomenuta okruženja u zadnjih par godina. Trenutno postoje veliki zahtjevi u industiji za aplikaciju modela dubokog učenja nad velikim skupovima podataka. Analogno tome, centralni problem postaje upravo problem skaliranja pomenutih aplikacija. Jedan od načina jeste distribuirano učenje modela na klasterima grafičkih kartica.

1.1 Distribuirano učenje modela

Postoje dve glavne filozofije u distribuiranom učenju modela. Paralelizam baziran na modelima i paralelizam baziran na podacima. U praksi se obično koristi paralelizam baziran na podacima zbog svoje jednostavnosti. On zahtjeva podjelu trening skupa i kloniranje modela ka svakoj grafičkoj kartici u klasteru i ažuriranje novodobijenih gradijenata nakon svake iteracije treninga. Samo ažuriranje gradijenata se može izvršiti preko takozvanog servera parametara [5] ili distribuirano upotrebom AllReduce [9] algoritma.

1.1.1 Server parametara

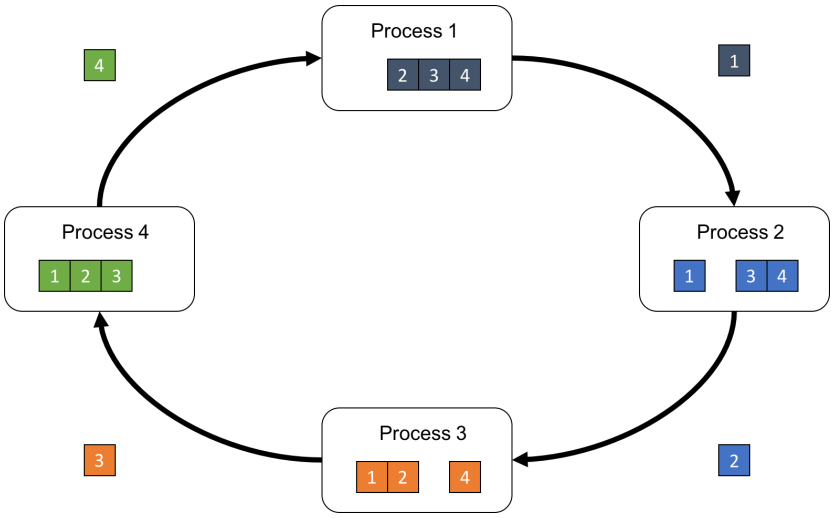
Nakon što jedan od radnika u klasteru završi sa treniranjem svog seta podataka, na serveru parametara je da primi ažurirane parametre i obavjesti ostatak klastera na promjenu modela. Ovaj pristup ima manu što sam server predstavlja usko grlo i jedinstvenu tačku otkaza 1. Takođe postoji problematika u kontekstu sinhronosti posla, odnosno da li čekati da se svi zadaci završe pa ažurirati parametre ili asinhrono ažurirati parametre po redoslijedu dolaska.



Slika 1: Ažuriranje parametara preko servera

1.1.2 AllReduce

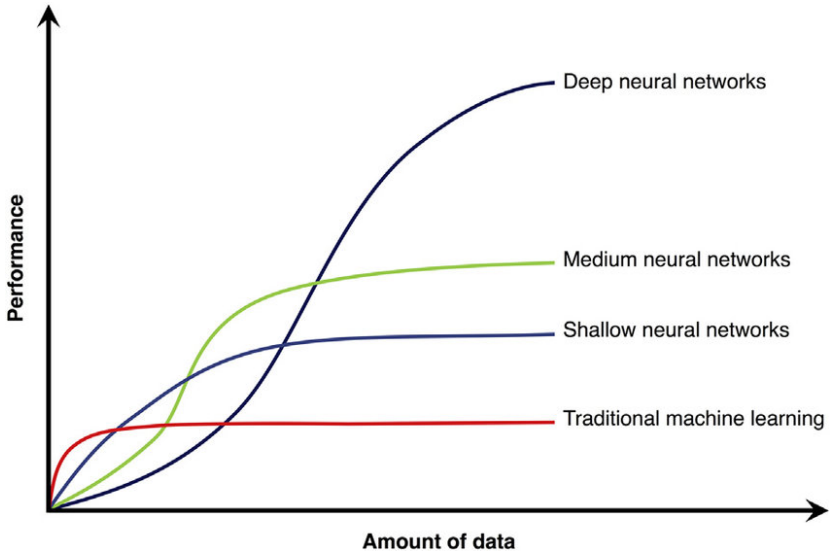
Za razliku od centralnog servera koji upravlja parametrima, upotreba Ring-AllReduce algoritma **??** omogućava distribuiran način ažuriranja parametara. Način na koji to postiže jeste taj da svaki od čvorova prosljeđuje dio kolekcije parametara sljedećem čvoru, koji vrši agregaciju sa svojim lokalnim parametrima. Postupak se ponavlja sve dok svi podskupovi parametara ne prođu kroz klaster.



Slika 2: Ažuriranje parametara preko AllReduce

2 Motivacija

Veliki napor se ulaže poslednjih godina u povećanje preciznosti dubokih mreža. Ono što duboke mreže omogućavaju u poređenju sa ostalim pristupima mašinskog učenja, jeste konzistentno povećanje preciznosti modela u odnosu na količinu trening podataka 3. Mnoge baze podataka za upoređivanje modela kao što su ImageNet [8] ili SQuAD [6], se sastoje iz eksplicitno označenih i preprocesiranih trening podataka, te kao takvi su zgodni za primjenu na postojećim okruženjima za duboko učenje. Ipak, većina izvora podataka u stvarnom svijetu su dinamički, neprocesirani i nestruktuirani. To podrazumjeva ozbiljnu pripremu tih podataka prije započinjanja bilo kakvih treniranja nad njima.



Slika 3: Odnos performansi raznih modela mašinskog učenja i količine trening podataka

Prva logična stvar jeste imati dva odvojena klastera za procesiranje podataka i drugi klaster za treniranje. Nažalost, analiza podataka u realnim aplikacijama je iterativan i rekurentan proces. U kontekstu debagovanja, razvoja i isporučivanja, bilo bi jako teško imati tako dve odvojene cjeline.

Jedan od načina da se adresira gore pomenuti problem jeste upotreba "povezivača", odnosno adaptera koji omogućavaju odgovarajuće interfejse za povezivanje različitih komponenti vezanih za obradu podataka i samog dubokog učenja. Ovaj pristup donekle omogućava cjelovitost aplikacije, ali i on nosi određene poteškoće. Tu problem predstavljaju razlike u međuprocesnoj komunikaciji, serijalizaciji podataka kao i razlike u impendansi modela (eng. impendance missmatch), koje adapter pokušava da premosti, naravno po cijeni performansi cjelokupnog sistema.

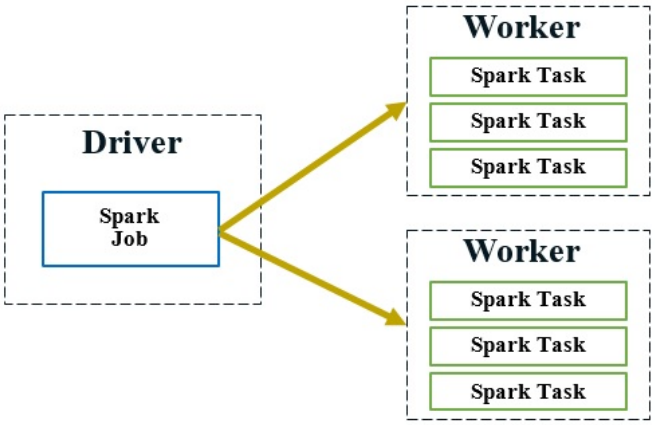
BigDL koristi drugačiji pristup koji podrazumjeva direktnu implementaciju distribuiranog dubokog učenja direktno na Apache Spark klasteru. To rezultuje eliminacijom razlike u impendansi modela i podređuje se modelu izvršavanja na Sparku.

3 BigDL model izvršavanja

Ključni problem koji je BigDL adresirao jeste implementacija funkcionalnosti dubokog učenja u kontekstu modela izvršavanja Sparka, koji je model krupne granularnosti (eng. coarse-grained). To znači da se operacije nad trening podacima i izračunavanje parametara izvršavaju nad čitavim skupovima podataka, odnosno kroz Spark transformacije. Takođe, podaci su nepromjenljivi (eng. immutable), što je velika razlika u odnosu na ostala okruženja dubokog učenja koje imaju promjenljiv (eng. mutable) pristup podacima zbog efikasnosti servera parametara pri distribuiranom treniranju.

3.1 Spark model izvršavanja

Spark klaster podataka sastoji se iz jednog glavnog(driver) čvora i više radnih(worker) čvorova 4. Glavni čvor je odgovoran za koordinaciju zadataka između više radnih čvorova u sklopu jednog Spark posla(Spark job), dok su radni čvorovi odgovorni za konkretno izvršavanje zadataka. Da bi se sam proces paralelizovao na nivou podataka, Spark podatke predstavlja u vidu RDD-a (eng. Resilient Distributed Dataset). RDD je nepromjenljiva kolekcija podataka particionisana unutar klastera i jedino se može transformisati u nove RDD kolekcije kroz funkcionalne operatore kao što su map, filter i reduce. Particionisanje RDD kolecija omogućava paralelnu primjenu operacija nad podacima u sklopu više Spark zadataka.



Slika 4: Spark model izvršavanja

3.2 Paralelno obučavanje BigDL modela na nivou podataka

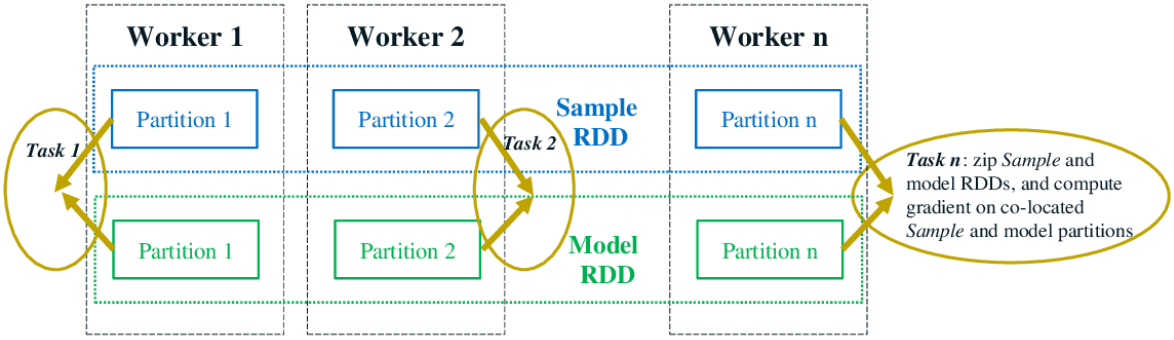
Distribuirano treniranje modela u BigDL je implementirano kao iterativan proces. 5 Svaka iteracija pokreće dva Spark posla, jedan za izračunavanje gradijenata, a drugi za ažuriranje parametara. Trening podaci su predstavljeni u vidu RDD Sample objekta, koji je automatski particionisan. Takođe, BigDL kreira dodatni RDD modela, gdje svaki model predstavlja kopiju originalnog modela neuronske mreže. Prije početka treniranja, ove dve RDD kolekcije su keširane u radnu memoriju i koparticionisane u zajedničke particije.

Algorithm 1 Data-parallel training in BigDL

```
1: for i = 1 to M do
2:     //"model forward-backward" job
3:     for each task in the Spark job do
4:         read the latest weights;
5:         get a random batch of data from local Sample partition;
6:         compute local gradients (forward-backward on local model
           replica);
7:     end for
8:     //"parameter synchronization" job
9:     aggregate (sum) all the gradients;
10:    update the weights per specified optimization method;
11: end for
```

Slika 5: Algoritam računanja parametara

Samo treniranje modela podrazumjeva primjenu zip operatora unutar analognih particija i zatim računanje lokalnih gradijenata u paraleli za svaki replicirani model. Bitno je napomenuti da BigDL ne podržava paralelizam zasnovan na modelu, usljed loših performansi tog pristupa.

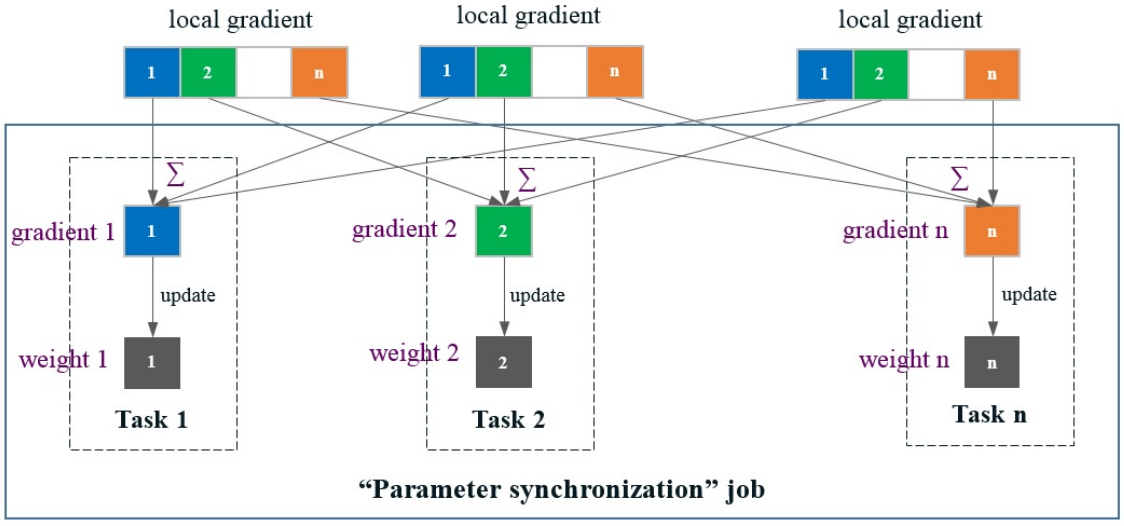


Slika 6: Paralelno računanje lokalnih gradijenata za svaku repliku modela

3.3 Sinhronizacija parametara u BigDL

Kao što je pomenuto ranije, sinhronizacija parametara je izuzetno bitna operacija unutar distribuiranog modela treniranja dubokih mreža. Stoga, svaki postojeći okvir za duboko učenje podržava server parametara ili AllReduce i to radeći operacije nad promjenljivim podacima. Ipak, ovakve operacije nisu podržane unutar sistema za obradu velikih količina podataka poput Sparka. BigDL rješava taj problem koristeći sljedeći algoritam 7:

- Nakon što je Spark posao sačinjen od N zadataka, odgovoran za izračunavanje lokalnih gradijenata završio posao, on izračunate gradijente ponovo ravnomjerno particioniše u novih N particija.
- Novi Spark posao je kreiran, čiji svaki n-ti zadatak ima za cilj upravljati svojom particijom lokalnih gradijenata. Prvo se razbacaju(eng. shuffle) odgovarajuće particije gradijenata u odgovarajući zadatak koji ih zatim agregira u vidu sume i s tim ažurira n-tu particiju težina
- Potom svaki zadatak uradi broadcast operaciju nad n-tom particijom tako da bi ostali taskovi mogli pročitati ažurirane težine prije početka sljedeće iteracije

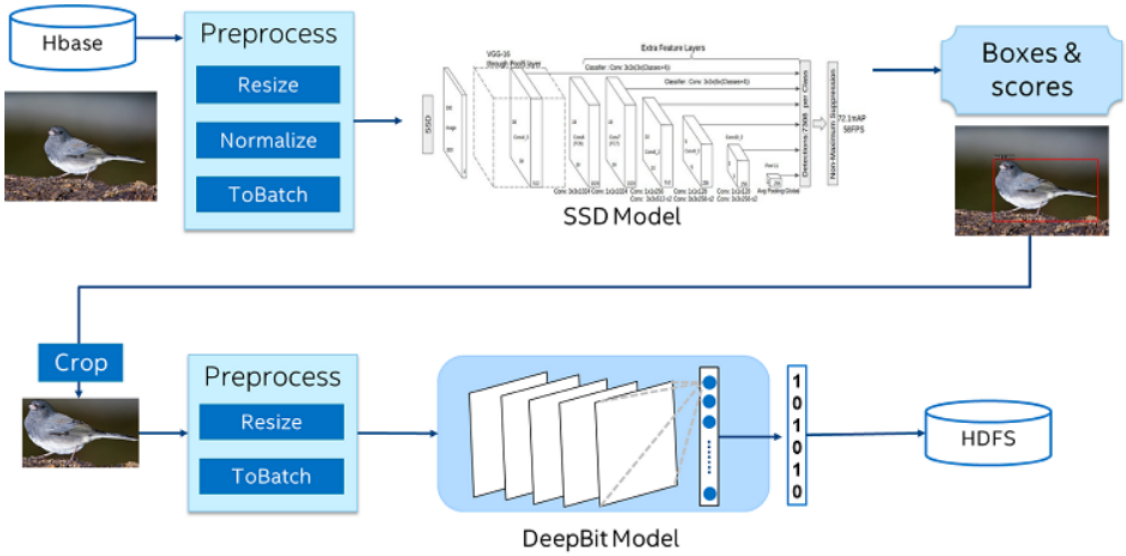


Slika 7: Prikaz sinhronizacije parametara za analogne particije

Ova implementacija AllReduce algoritma u BigDL-u ima slične performanse kao i Ring AllReduce u okruženjima za duboko učenje. Takođe, opseg svakog čvora je maksimalno iskorišćen i komunikacija između čvorova je samim tim optimizovana. Kao rezultat toga, BigDL može efektivno izvršavati treniranje nad velikim klasterima, što garantuje skalabilnost.

4 Slučajevi korištenja BigDL okruženja

Izdvajanje obilježja iz slika je jedna od glavnih tema u današnjoj primjeni dubokog učenja. Koristi se od pretrage sličnosti slika, do uklanjanja duplikata odnosno deduplikacije slika. Kompanija JD.com, jedna od najvećih kompanija za online prodaju, uzela je za zadatak da omogući efikasno izdvajanje obilježja iz ogromne količine slika njihove robe. Iako su detekcija objekata i izdvajanje obilježja iz slika prilično standardni algoritmi kompjuterskog vida (eng. computer vision), kada se ti problemi skaliraju na stotine miliona slika u produkciji, počinje se vidjeti značajan porast kompleksnosti.

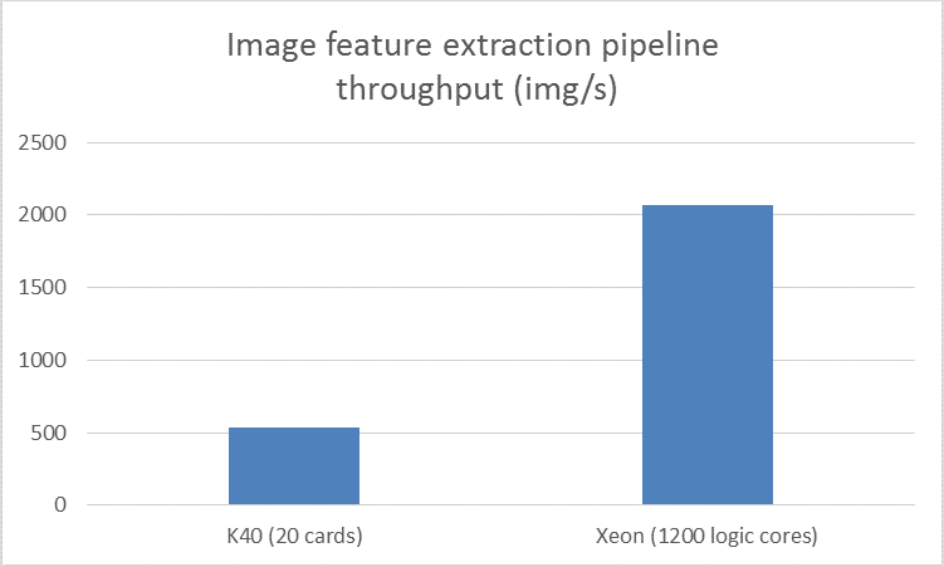


Slika 8: Slučaj korištenja BigDL okruženja u sklopu izdvajanja obilježja slika

Prethodno, inžinjeri zaposleni u JD.com, pokušali su adresirati ovaj problem pokrećući treniranje modela na GPU klasteru od 5 čvorova koristeći princip adaptera: čitajući podatke iz HBase baze podataka, particionisajući i procesirajući podatke kroz klaster i zatim pokretati treninge modela na Caffe okruženju. Neki od problema koji su se pojavili usljed ovog pristupa su:

- Čitanje podataka iz baze traje mnogo vremena u poređenju sa radom grafičkih kartica i samo čitanje ne može lako biti optimizovano na GPU klasteru
- GPU klaster nije optimizovan za preprocesiranje slika
- Programeri moraju uložiti mnogo truda u ručno particionisanje podataka i balansiranje zadataka
- Loša podrška za otpornost na otkaze
- Aplikacije bazirane na GPU klasterima, imaju mnogo zavisnosti, kao npr. CUDA, koja uvodi značajan nivo kompleksnosti u produkcioni sistem i operacije

Da bi prevazišli gore navedene probleme, JD inžinjeri su implementirali BigDL okruženje i napravili rješenje koje predstavlja jedan jedinstveni cjevovod, uključujući učitavanje podataka, particionisanje podataka, preprocesiranje i treniranje modela 8. Implementacija je pokrenuta na jednom Spark klasteru kao distribuiran problem. Poboljšanje u brzini izračunavanja i izdvajanja obilježja u poređenju sa prethodnim rješenjem je čak približno 3.83 puta 9.



Slika 9: Poređenje protočnosti obrade slika K40 GPU i Intel Xeon procesora

5 Zaključak

Visoka skalabilnost, visoke performanse i lakoća korištenja BigDL okruženja, definitivno omogućava rješenja mnogih problema u industriji vezanih za obučavanje dubokih mreža. Vjerovatno u budućnosti, mnoge firme pokušaće koristiti ili nadograditi postojeće BigDL okruženje, prateći trend porasta trening skupova podataka.

Literatura

[1] Berkeley. Caffe. https://caffe.berkeleyvision.org/.

[2] Facebook. Pytorch. https://pytorch.org/.

[3] Google. Tensorflow. https://www.tensorflow.org/.

[4] Xin Qiu Jason Dai, Ziheng Wang. BigDl: A distributed deep learning framework for big data. *ACM Symposium of Cloud Computing conference*, 2019.

[5] Mu Li; Li Zhou; Zichao Yang; Aaron Li. Parameter server for distributed machine learning. *Carnegie Mellon University, Google Strategic Technologies.*

[6] Pranav Rajpurkar; Jian Zhang; Konstantin Lopyrev; Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

[7] Michael J. Franklin Matei Zaharia, Mosharaf Chowdhury. Spark: Cluster computing with working sets. *University of California, Berkeley*, 2014.

[8] Jia Deng; Wei Dong; Richard Socher. Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[9] Pitch Patarasuk; Xin Yuan. Bandwidth optimal all-reduce algorithms for clusters of workstations. *Department of Computer Science, Florida State University.*