Merenje performansi

Veljko Petrović
Oktobar, 2022

Zatečeno stanje

Šta je to što imamo?

Commodity Cluster

- Dosta je priče o tome kako računari možda izgledaju i kako su nekad izgledali.
- Kako izgleda super-računar danas?
  - Ključnije: Kako izgleda super-računar koji *mi* koristimo.

Arhitektura ugrubo

- Imamo više čvorova koji su povezani nekakvom mrežom.
- Svaki čvor ima:
  - n fizičkih procesora koji imaju
  - m logičkih procesora (jezgara) koji podržavaju
  - k niti izvršavanja
  - neku količinu l memorije koji ti procesori dele
  - g kartica za ubrzanje sa svojim specijalizovanim resursima za proračunavanje
- Ti čvorovi su povezani u jedan sistem kroz nekakvu mrežu

U praksi...

- Računari sa kojima mi radimo će gotovo sigurno biti commodity cluster tipa.
- To znači 1 čvor = 1 PC
- Taj neki čvor ima:
  - 1 procesor
  - 4-32 jezgara koji podržavaju po
  - 2 niti izvršavanja
  - Neki broj GB memorije (~16GB)
  - jednu ili dve grafičke kartice tipično NVidia tipa
- Svi čvorovi su povezani na brzu Ethernet mrežu.

Tipično NVidia?

- Ovaj predmet ne sponzoriše NVidia
- Razlog što su NVidia kartice tipične se vrti oko toga što, kada je u pitanju naučna primena GPU uređaja, NVidia je stigla prva.
- Kao rezultat toga što je to prvo počelo na NVidia karticama, dosta softvera je napravljeno sa tom idejom na umu
- Danas, polako, svi proizvođači kartica počinju da podržavaju naučni rad
- Najbrži računar, na primer, je napajan AMD-om

U praksi...

- Svaki računar je individualna mašina, sa svojom instalacijom Linux-a.
- Na svakom računaru su instalirani određeni alati, biblioteke, itd. koje omogućavaju da se resursi između računara efektno iskoriste.
- Primer:
  - OpenMP nam omogućava da koristimo paralelizam na nivou te jedne mašine.
  - OpenMPI nam omogućava da koristimo paralelizam između mašina.
  - OpenACC nam omogućava da koristimo GPU resurse na udoban način.
    - Alternative su CUDA ili OpenCL
  - SLURM nam omogućava da kontrolišemo šta se izvršava i gde.

Naš plan

- Da bi savladali ovo, naš plan jeste da:
  - **Benchmarking.** Naučimo kako merimo i biramo super-računarsku instalaciju.
  - **Resource** management. Naučimo kako da upravljamo onim što imamo.
  - **Parallel programming.** Naučimo tehnologije koje nam omogućavaju paralelizme.
  - **The Problem.** Savladamo neke primere problema sa kojima se suočavamo i smislimo kako da ih ubrzamo.
  - **Domain specific libraries.** Naučimo kako da koristimo specijalizovane biblioteke za istu svrhu.
  - **Profiling.** Naučimo kako da izmerimo to što smo napravili.
  - **Visualization.** Naučimo kako da prikažemo šta smo izračunali.

Merenje performansi HPC sistema

Koliko je brzo ‘brzo’

Koncept

- Sabiranje performansi konstituenata HPC sistema proizvodi nekakve brojeve: toliko-i-toliko FLOPS-a.
- To nisu osobito korisni brojevi. Mogli bi i da sabiramo težine komponenti: koja je svrha ako nam ta vrednost nešto ne kaže.
- Možemo da modeliramo ponašanje, ali taj model bi uskoro postao izuzetno nezgodan za korišćenje i opet ne bi uhvatio sve detalje ponašanja.
- Rešenje je jednostavno: postavimo nekakav problem i merimo koliko je naš sistem dobar u njegovom rešavanju.
- Ako razmislite o tome, ovo je isto što se radi studentima:
  - Postavi se problem (ispit)

- I mere se performanse (ocena).

Problemi

- Naravno, ni ovo nije savršeno.
- Kakav tip problema se izabere, kako se postavi, konfiguriše, i koristi jako utiče na rezultat koji se dobije.
- Grubo gledajući, možemo da podelimo sva merenja performansi na:
  - Sintetičke
  - Prirodne
- Sintetički se lako kontrolišu i skaliraju.
- Prirodni daju izuzetno verodostojne rezultate—samo što nismo sigurni šta je to što mere.
- Primer van HPC: 3DMark vs. framerate.

Malo istorije...

- Prvi benchmark ikada je bio za ENIAC (prikladno) i bio je računanje trajektorije đuleta u odnosu na isti proračun koji obavlja balistički računar ili ljudsko biće.
- Inženjer je, na kraju krajeva, prvobitno bio vojni termin.
- Prvi benchmark u široj upotrebi je ‘Whetstone’ nazvana po gradiću u kome je razvijena.
- Whetstone benchmark je bila kolekcija programa koji su stvarali sintetički problem koji je evaluirao broj instrukcija u datoj sekundi. Kasnije, kako su se potrebe menjale, počeo je da uključuje i floating-point instrukcije, te je merio i FLOPS.

Malo istorije...

- Pošto je Whetstone specijalizovan za FLOPS, napravljen mu je pandan koji meri performanse sa celim brojevima.
- A pošto programeri vole malo šta više od igre rečima, ovaj pandan se zvao ‘Dhrystone.’
- Danas Dhrystone je zamenjen sa SPECint paketom.
- I Whetstone i Dhrystone nisu bili namenjeni za HPC, već za evaluaciju procesora.

Linpack

- Najuticajniji benchmark u upotrebi se pojavio 1979 i baziran je na bibliotekama za linearnu algebru poznate kao Linpack (danas zamenjene sa Lapack/BLAS paketom softvera).
- Linpack meri performanse floating-point operacija, i baziran je na rešenju problma oblika:

Linpack

$$Ax = B$$

Gde je A $n \times n$ matrica, x je vektor $x_0, x_1, \ldots x_{n-1}$, b je vektor $b_0, b_1, \ldots b_{n-1}$ **Ograničenje:** A mora da ima malo ili nimalo elemenata koji su jednaki ili blizu nule, još se kaže i A je *gusta* matrica. Ovo je efektivno zadatak rešavanja sistema od n jednačina sa n nepoznatih.

Linpack

- Linpack je prošao kroz iteracije:
  - Prva iteracija je stavljala n na 100 i algoritam koji je koristila je bio serijski. Da bi rezultat bio merodavan, izvorni kod nije smeo da bude modifikovan: jedino parametri kompajliranja.
  - Druga iteracija je podigla n na 1000 i:
    - Dozvolila je da se prilagođava izvorni kod faktorizatora i rešavača.
    - Dozvolila je, konsekventno, paralelizaciju.
    - Uvela je uslov tačnosti nad rezultatima.
- Treća iteracija je HPL.

Highly Parallel Linpack

- Highly Parallel Linpack — HPL je naznačajniji benchmark svoje vrste. On, pre svega, se koristi da meri FLOPS vrednosti za super-računare i služi da se lista 500 najbržih odredi.
- HPL je uveo još značajnije promene:
  - Samo je problem fiksan
  - Veličina ulaznih podataka može da varira.
  - Softver može da varira
- Ove promene dozvoljavaju HPLu da se izvršava na uređajima sa distribuiranom memorijom.

Parametri HPL-a

- Do 20 različitih zadataka je dozvoljeno, mi specificiramo koliko hoćemo da probamo.
- Svaki od zadataka je specificiran preko ‘n,’ tj. broja jednačina i broja nepoznatih.
- Zatim je dozvoljeno do 20 različitih veličina bloka
  - Bloka? Način na koji algoritmi ovog tipa rade jeste da rekurzivno razbijaju matricu na pod-matrice. Blok definiše koliki će biti individualni segment.
  - Gotovo sigurno je u rasponu 32..256 a određuje se empirijski.
- Zatim se specificira kako se matrica deli po čvorovima
- Dalje, kaže se koliko različitih konfiguracija procesnih čvorova se koristi za proračun.

Malo više o blokovima

- Neka je naš zadatak da odredimo $C = AB$ gde su A, B, i C 8×8 matrice.
- Onda možemo da izdelimo sve te matrice u 4 4×4 podmatrice.

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Malo više o blokovima

Gde je, npr:

$$A_{i,j} = \begin{bmatrix} a_{i,j} & a_{i,j+1} \\ a_{i+1,j} & a_{i+1,j+1} \end{bmatrix}$$

Malo više o blokovima

A mi računamo:

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$

$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$

$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$

$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

Malo više o blokovima

- Onda su ove pod-matrice su naši blokovi
- Naravno, umesto 4×4 to su, recimo, 256×256
- U praksi, ovo je ideal: nisu sve matrice.
- Veličine i distribucije ovih blokova se stalno menjaju i podešavaju
- Ideja je da se promoviše u kodu osobina *lokalnosti*
- Setite se situacije sa kešom

Parametri HPL-a

- Zatim se specificiraju sve konfiguracije, jedna po jedna, u obliku $P \times Q$
- Zašto tako? Zato što se matrica bukvalno deli među čvorovima tako što se čvorovi zamisle u matričnoj organizaciji, te se svakom čvoru “odseče” određeni segment matrice.
- Ovo znači da će naš proces biti pokrenut, nužno, na $P \times Q$ čvorova, šta god drugo mi specificirali prilikom pokretanja.
- P i Q se biraju empirijki i zavise od topoloigije naše mreže. Ako je mreža u mesh topologiji tako da imamo više linija za svaki računar onda je najbolje da je P blizu Q a Q malo veće od P. U slučaju prostog Ethernet-a najbolje je biti pljosnat: Za klaster od 4 računara, recimo, 1×4

Parametri HPL-a

- Definiše se još i stepen tačnosti do kog računamo. `16.0` je tipična vrednost, mada se ona može povećavati i do stotina hiljada. Ovo je zbog nepreciznosti poređenja floating point brojeva, prvo, i skale koja se koristi za test.
- Ostatak služi za podešavanje algoritma o čemu više kada budemo radili linearnu algebru u HPC okruženju.

Rezultati HPL-a

- HPL nam omogućava da odredimo veći broj vrednosti i od njih nas zanima:
  - `Rpeak` — Broj FLOPSa koji smo izračunali da je teoretski moguć
  - `Rmax + Nmax` — Ovo je veličina ulaznog skupa (`Nmax`) za koju je ostvaren najveći stvarni broj FLOPSa (`Rmax`).
  - `N1/2` — Veličina ulaznog skupa za koju je ostvareno pola maksimalne ostvarene performanse.

Osobine dobrog benchmark sistema

- Da li je HPL dobar benchmark? Šta znači biti “dobar” u ovom kontekstu?
- Bazirano na iskustvu može se reći da dobar benchmark ima sledeće osobine:
  - **Realizam.** Mora simulirati tip rada na koji se nailazi u praksi.
  - **Univerzalnost.** Adekvatno radi na velikom broju arhitektura.
  - **Popularnost.** Dosta korisnika znači da imamo dosta uporedivih rezultata.
  - **Kompaktnost.** Mali broj linija koda znači manje šuma.
  - **Uređenost.** Postoje jasna pravila oko toga kako se dobijaju merodavni rezultati.
  - **Standardizovanost.** Upotreba standardnih tehnologija.

HPC Challenge

- HPC Challenge Benchmark Suite je malo opširnija alternativa HPL sistemu.
- Sa pozitivne strane: meri mnogo više stvari.
- Sa negativne strane: ne proizvodi jednu stvar, nego više, što čini samerljivost kompleksnijom. Čak i da primenimo matematičku koncepciju razdaljine u višedimenzionalnom prostoru, i dalje moramo da ustanovimo zajedničku skalu plus da odredimo funkciju metrike. Kao rezultat moguće je formulisati beskonačno mnogo samerljivosti što nas ostavlja sa ogromnim koordinacionim problemom.

HPC Challenge

- HPCC se sastoji od sledećih mera:
  - `DGEMM`
    - Množenje matrica * matrica, slično HPL u nameni
  - `STREAM`
    - Sintetički problem koji meri dugoročnu protočnu moć memorije
  - `PTRANS`
    - Paralelizovano transponovanje matrica
  - `RandomAccess`
    - Nasumično postavlja nove celobrojne vrednosti u nasumične delove memorije. Meri se u Giga Updates Per Second: GUPS.
  - `FFT`
    - Jednodimenzionalna kompleksna diskretizovana Furjeova transformacija
  - `B_eff`
    - Kašnjenje i protočna moć za različite obrasce komunikacije

Šta odlikuje HPCC?

- Fokus na memoriju.
- Distribuirani sistemi su izuzetno osetljivi na zadatke koji zahtevaju slobodan tok podataka kroz ceo sistem.
- HPCC stress-testira baš ovaj deo koji konvencionalniji HPL uglavnom ignoriše.
- Ovo ne čini HPL-lošim niti HPCC dobrim.
  - Što?
  - **Namena računara diktira šta želimo da merimo.**

High Performance Conjugate Gradients

- HPCG održavaju isti ljudi kao i HPL i služi da meri stvari koje HPL ne dodiruje.
- Problem koji HPC rešava je sasvim isti kao onaj koji rešava HPL
- Pa koja je onda razlika?

HPCG

Rešavamo problem:

$$Ax = B$$

Gde je A $n \times n$ matrica, x je vektor $x_0, x_1, \ldots x_{n-1}$, b je vektor $b_0, b_1, \ldots b_{n-1}$ **Ograničenje:** Ali sada A ima **većinu** elemenata koji su 0 ili vrlo blizu nula, odnosno A je *retka* matrica.

HPCG

- To što je A retka matrica menja tehnologiju za rešavanje potpuno.
- Glavna je stvar to što ima puno komunikacije između čvorova i puno redukcionih operacija o čemu više uskoro.
- Ovo menja rezultate jako, jako, jako puno.
- Koliko puno?
  - Razlika između ostvarenih performansi za HPL i HPCG je 15000%.
  - Prosečno. Vrhunski sistemi ostvaruju na HPCG oko 1% onoga što ostvaruju na HPL
  - Postoje izuzeci, ali čak i šampion, tkzv. K-računar ostvaruje samo 5.3%.

NAS Parallel Benchmarks

- NPB je sličan HPCC sistemu u tome što predstavlja kombinaciju tipičnih problema.
- Specifično predstavlja 8 tipičnih problema koji su poznati pod svojim dvoslovnim skraćenicama.

NPB

- **IS** (Integer Sort) - Sortiranje celih brojeva. Testira celobrojnu brzinu i performanse mreže.
- **EP** (Embarassingly parallel) - Testira brzine blizu teoretskog maksimuma.
- **CG** (Conjugate Gradient) - Radi račun karakteristične vrednosti retke matrice.
- **MG** (MultiGrid) - Integracija diferencijalne jednačine kroz metodu više rezolucija.

NPB

- **FT** (Discrete 3D FFT) - Integracija parcijalne diferencijalne jednačine kroz brze Furijeove transformacije
- **BT** (Block Tridiagonal Solver) - Rešavanje sistema jednačina trodijagonalnog tipa.
- **SP** (Scalar Pentadiagonal Solver) - Rešavanje sistema jednačina pentadijagonalnog tipa.
- **LU** (LU Gauss-Seidel solver) - Isto što i HPL efektivno

Pentadijagonalno? Tridijagonalno?

- HPC *mnogo* voli matrice te je korisno da znate terminologiju
- *Retka* matrica je takva da joj je većina elemenata nula.
- *Trakasta* matrica je retka matrica takva da su joj nenulti elementi isključivo na glavnoj dijagonali i na nekim dijagonalama iznad glavne i ispod glavne, ili formalno govoreći:
  $a_{i,j} = 0$ ako $j < i - k_1$ ili $j > i + k_2$ za $k_1, k_2 \geq 0$

Trakaste matrice

- Vrednosti k_1 i k_2 su gornja i donja *širina trake*
- Ako $k_1 = k_2$ onda je ovo simetrična trakasta matrica, i obično se zove n-dijagonalna, po tome koliko ukupno dijagonala ima popunjenih ili delimično popunjenih.
- Stoga:
  - Ako je $k_1 = k_2 = 1$ onda je tridijagonalna matrica u pitanju
  - Ako je $k_1 = k_2 = 2$ onda je pentadijagonalna matrica u pitanju
  - Ako je $k_1 = k_2 = 3$ onda je heptadijagonalna matrica u pitanju
  - ...

Graph500

- Graph500 ne samo da odgovara tom domenu nego radi i u širem kontekstu—simulira aplikacije sa intenzivnim pristupom podacima.
- Graph500 definiše, u stvari, 3 benchmark-a
  - Graph500 1 — konkurentna pretraga
  - Graph500 2 — Najkraća putanja
  - Graph500 3 — Najveći nezavisni skup (tj. najveći skup čvorova grafa takvih da nikoja dva nisu povezana)
- Samo Graph500 1 je implementiran i to preko Breadth First Search pristupa, tako što za 64 jedinstvena početna čvora u tom grafu nađe sve čvorove do kojih se može doći.

Graph 500

- Do ovog trenutka, postoji određena tendencija u svim ovim merama:
  - Diferencijalne jednačine.
  - Sistemi linearnih jednačina raznih tipova.
  - Bazične celobrojne operacije.
- Ovo su primeri problema simulacija za građevinu, mašinstvo, i fiziku, opšte govoreći.
- To je lepo, ali nisu svi problemi ovakvi.
- Veliki skupovi problema se predstavljaju kroz operacije nad grafovima.

- Graf je cikličan, bez težina, i neusmeren.

- Ovo možete čitati kao “breakthrough wanted” znak.

Graph500

- Graph500 je implementiran sa istovremenim OpenMP i OpenMPI paralelizmom što odgovara modernim sistemima.
- Rezultate ne vraća u FLOPS nego u TEPS (Traversed Edges Per Second)
- Poređenje između FLOPS i TEPS je teško ali ima jedna druga stvar koja je *jako bitna:*
  - U FLOPS-ima računari nastavljaju svoj eksponencijalni rast.
  - U TEPS-ima performanse stagniraju.
  - Arhitekture deljene memorije su mnogo bolje (u kontekstu TEPS po jezgru) od arhitektura distribuirane memorije.

Miniaplikacije

- Sve do sada, svi ovi benchmark-ovi rade sa sintentičkim podacima koji pokušavaju da simuliraju šta bi stvarna aplikativna primena postigla.
- Mogu se meriti rezultati punih aplikacija, naravno, ali su oni tako specifični da nisu osobito korisni: nisu univerzalni i nisu sveprisutni.
- Pokušaj da se odradi kvadratura kruga ovde jesu miniaplikacije: pojednostavljeni ali realni poslovi.
- Ovoga se dotičemo kada budemo radili domenski-specifične biblioteke mnogo, mnogo kasnije.