Alokacja zadań w chmurze

Tomasz Kanas

22 października 2020

Wprowadzenie

- Klaster obliczeniowy składa się z wielu maszyn (w Google mediana to ok. 10 000)
- Każda maszyna posiada określone zasoby (n.p. CPU, RAM, przepustowość sieci)
- Na każdej maszynie wykonuje się jednocześnie wiele zadań (rząd wielkości: 30)

Problem:

Jak przydzielać zadania do maszyn?

Możliwe korzyści:

- mniejsze zużycie prądu: datacenter potrafi zużywać nawet 100 MW (tyle co ok. 250 000 domów)
- Oszczędność na sprzęcie

Roadmap

1. Rozwiązania używane w Google
   - Borg: Scheduler Google'a
   - Autopilot: Automatyczne dostosowywanie limitów
2. Formalizacja i teoria
   - Algorytmy aproksymacyjne
   - Dane statystyczne i przewidywanie obciążenia
   - Wybrane dowody

Borg

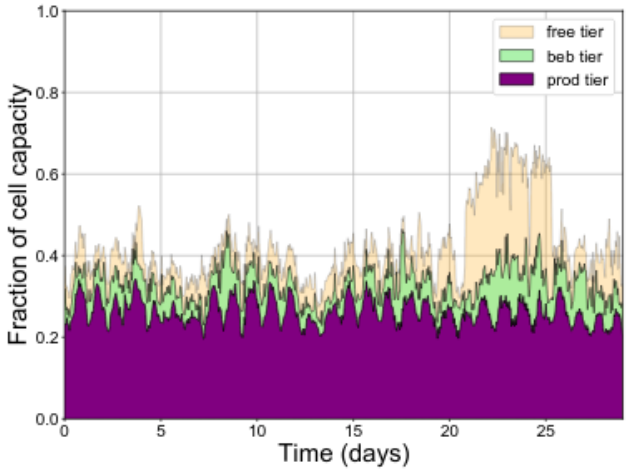
Dane wejściowe

- binarka
- liczba replik
- CPU (soft cap)
- RAM (soft lub hard cap)
- priorytet:
  - Free tier: najniższy, brak SLOs (Service Level Objectives)
  - Best-effort Batch tier: wypełniacze, brak SLO, batch scheduler
  - Mid-tier: mało wymagające SLO
  - Production tier: wysokie SLO, przerywanie niższych tierów aby zwolnić zasoby
  - Monitoring tier: kluczowe dla infrastruktury

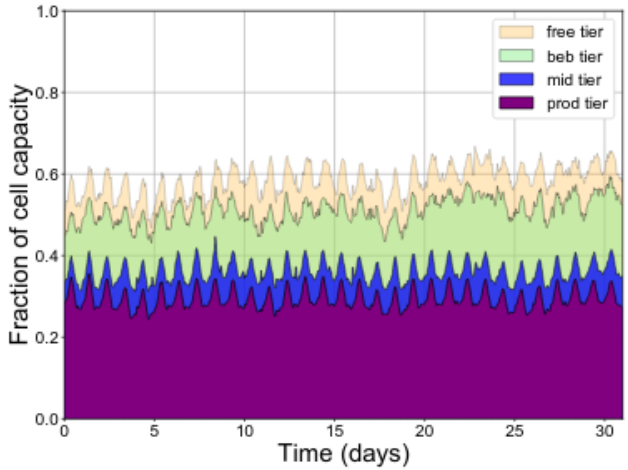
Borg — planowanie

- Master scheduler na klaster + local scheduler na maszynę
- Master przydziela zadania do maszyn
- Zwykle alokuje ponad 100% na maszynę (użytkownicy zawyżają wymagania)
- Może przerwać i ponownie zaplanować zadanie aby:
  - zainstalować aktualizacje
  - zwolnić miejsce dla ważnych zadań
  - planować bardziej efektywnie
- Autoscaling: zadanie może dynamicznie zażądać więcej replik lub wyższe limity (co może doprowadzić do przerwania zadań o niższym priorytecie)

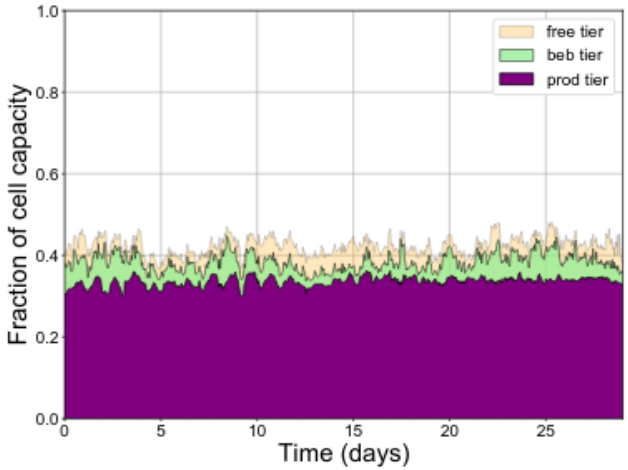
Borg — użycie klastra



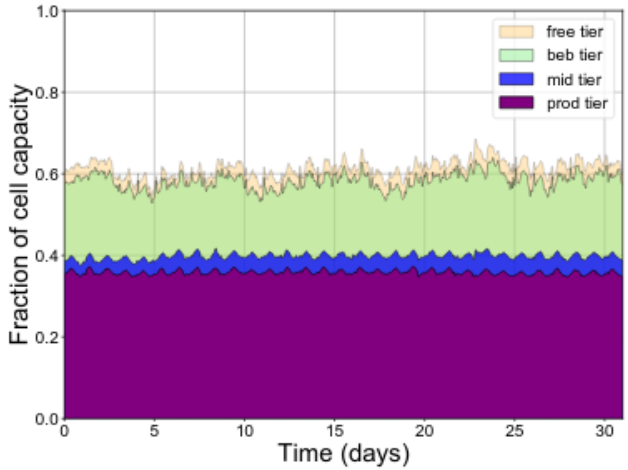
(a) 2011 CPU usage.



(b) 2019 CPU usage, averaged across all 8 cells.



(c) 2011 memory usage.



(d) 2019 memory usage, averaged across all 8 cells.

Wariancja wymagań zadań

Współczynnik C^2

$$C^2 = \text{wariancja}/\text{wartość oczekiwana}$$

Niezmienniczy wzgl. skalowania danych, proporcjonalny do średniej długości oczekiwania zadania.

- Google, CPU: $C^2 = 23000$, RAM: $C^2 = 43000$
- Akamai (USA, 2017) $C^2 = 760$
- rozkład wykładniczy: $C^2 = 1$

W Google 1% zadań odpowiada za ponad 99% obciążenia maszyn.

Autopilot: automatyczne skalowanie zadań

- System do automatycznego dostosowywania wymagań zadań w Borg-u
- Dostosowuje wymagania CPU i RAM aby zminimalizować różnicę między limitem a rzeczywistym zużyciem (vertical scaling)
- Dostosowuje liczbę replik do większych zmian obciążenia (horizontal scaling)
- Pobiera dane o obciążeniu z niezależnego systemu monitorującego
- Wysyła wyniki do Borgmastera, który aplikuje zmiany (istniejący autoscaling)

Autopilot — dane wejściowe

- $r_i(\tau)$ — wartość pomiaru dla repliki i w czasie τ. Pomiary zwykle co 1 sekundę.
- Aggregowane do histogramów, wartość kubełka k (zwykle $k \leq 400$) w oknie czasowym t (zwykle 5 min)
  $s_i[t][k] = |\{r_i(t) : \tau \in t \wedge b[k-1] < r_i[\tau] < b[k]\}|$,
  $b[k]$ — granice kubełków
- Sumujemy histogramy dla wszystkich replik
  $s[t][k] = \sum_i s_i[t][k]$
- skalowanie danych ze względu na wiek τ, waga:
  $w[\tau] = 2^{-\tau/t_{1/2}}$, $t_{1/2}$: stała (12h dla CPU, 48h dla RAM), czas po którym waga spada o połowę

Autopilot — vertical scaling: statystyki

- Największe obciążenie wśród ostatnich N próbek

$$S_{max}[t] = \max_{\tau \in \{t-(N-1),\ldots,t\}} b[j] : s[\tau][j] > 0$$

- Średnie ważone obciążenie

$$S_{avg}[t] = \frac{\sum_{\tau=0}^{\infty} w[\tau]\overline{s[t-\tau]}}{\sum_{\tau=0}^{N} w[\tau]}$$ gdzie $\overline{s[\tau]}$ to średnia histogramu

- j-ty percentyl dostosowanego (uśrednionego po wartości i wadze) obciążenia

$$h[t][k] = b[k] \sum_{\tau=0}^{\infty} w[\tau] s[t-\tau][k], \quad S_{pj}[t] = P_j(h[t])$$

Autopilot — vertical scaling

CPU

- Batch jobs: S_{avg} — przy braku SLO najwyższa wydajność
- Pozostałe: S_{p95} lub S_{p90} w zależności od wymagań

RAM

- Niska tolerancja (na OOM — Out Of Memory Error): S_{p98}
- Minimalna tolerancja: S_{max}
- Średnia tolerancja: $\max(S_{p60}, 0.5 \cdot S_{\max})$

Post-processing

- Zwiększenie o 10% - 15% (dla bezpieczeństwa)
- Wzięcie maksimum z ostatniej godziny

Autopilot — vertical scaling: machine learning

- Wiele, prostych (2 parametry) modeli (wyniki powinno dać się wyjaśnić)
- Każdy model liczy koszt danego limitu używając ostatniego histogramu, na podstawie:
  - Liczby zadań poniżej limitu (zbyt małe użycie)
  - Liczby zadań powyżej limitu (nadużycie)
  - Faktu zmiany limitu (może skutkować przeschedulowaniem)
- Wybierany jest najlepszy model, biorąc pod uwagę:
  - koszt wyniku modelu
  - fakt zmiany limitu
  - fakt zmiany modelu

Autopilot — horizontal scaling

Rekomendacja — do wyboru

- Utrzymanie docelowego obciążenia CPU na poziomie r^*, za pomocą statystyki S i danych z horyzontu długości T

$$n_r[t] = r_S[t]/r^* \text{ gdzie } r_S[t] = S_{\tau \in [t-T,t]} \sum_i r_i[\tau]$$

- Funkcja dostarczona przez użytkownika

Stabilizacja — do wyboru

- Max z ostatnich T godzin (T podane przez użytkownika)
- Rozłożenie w czasie zbyt dużych spadków liczby zadań
- Ignoruje zbyt małe różnice
- Limit (procentowy) na liczbę nowych zadań

Autopilot — rezultaty

- Spadek względnej różnicy między wymaganiami a zużyciem z 60% dla twardych limitów i 46% dla miękkich limitów do 31% dla statystyk i 23% dla ML.
- Spadek liczby OOM (Out Of Memory Error)
  - Bez autopilota, twarde limity: 98,7% zadań bez OOM, 0.069 OOMs / zadanio-dzień
  - Bez autopilota, miękkie limity: 97.5% zadań bez OOM, 0.19 OOMs/ zadanio-dzień
  - Autopilot, statystyki: 99.5% zadań bez OOM, 0.002 OOMs / zadanio-dzień
  - Autopilot, ML: 0.013 OOMs/ zadanio-dzień

Postawienie problemu

Dane wejściowe

- M maszyn
- Z zasobów
- N zadań
- Maszyna m posiada z_{mz} zasobu z
- Zadanie n wymaga w_{nz} zasobu z

Cel

Znaleźć przydział zadań do maszyn, taki, że:

- żadna maszyna nie jest przeciążona
- jak najwięcej zadań jest przypisanych
- średnie obciążenie używanych maszyn jest maksymalizowane
- używanych jest minimalna liczba maszyn

Rozważane są też inne sformułowania, w szczególności często rozważa się wersje online.

Uproszczenie i formalizacja

Uproszczenia

- Jeden zasób
- Takie same maszyny
- Nieskończona liczba dostępnych maszyn
- Minimalizujemy liczbę potrzebnych maszyn

Sformułowanie

Znaleźć przydział zadań do maszyn, tj. odwzorowanie $F : \{1, \ldots, N\} \to \mathbb{N}$ spełniające

$$\forall_{m \in \mathbb{N}} \sum_{n \in F^{-1}(m)} w_n \leq z$$

Oraz minimalizujące

$$\max_{n \in F(\{1,\ldots,N\})} F(n)$$

Jest to znany NP-trudny problem pakowania (bin packing problem).

Niedeterminizm

Często bardzo trudno oszacować ile zasobów wymaga dane zadanie, zwykle oszacowania zawyżają zapotrzebowanie „na wszelki wypadek".

Modyfikacja: Stochastic bin packing

Zadanie n będzie wymagało X_n zasobów, gdzie X_n jest zmienną losową. Dla uproszczenia zakładamy, że X_n są niezależne i nie zależą też od czasu. Wymaganiem jest teraz:

$$\forall_{m \in N} \mathcal{P}\left(\sum_{n \in F^{-1}(m)} X_n > z \right) < \eta$$

Gdzie η jest pewną stałą zależącą zwykle od potrzeb jakości usługi (QoS: Quality of Service)

Algorytmy aproksymacyjne

Algorytm	Opis	Aproks.
Next-fit	Jeden otwarty kubełek, jeśli nowy przedmiot się w nim nie mieści to zamykamy obecny kubełek i otwieramy nowy	2
First-fit	Wrzuca przedmiot do dowolnego kubełka w którym ten się mieści	1.7
Best-fit	Wrzuca przedmiot do najbardziej zapełnionego kubełka w którym ten się mieści	1.7
First-fit-decreasing	First-fit z przedmiotami posortowanymi malejąco (offline)	11/9

Uwaga

Żaden algorytm online nie może aproksymować lepiej niż 1.5

Next-fit (NF)

Jeden otwarty kubełek, jeśli nowy przedmiot się w nim nie mieści to zamykamy obecny kubełek i otwieramy nowy

Dowód: 2-aproksymacja

Średnie zapełnienie ostatno zamkniętego i nowo otwartego kubełka $> 50\% \implies$
średnie wypełnienie wszystkich kubełków (ew. poza ostatnim) $> 50\% \implies$
nie da się zmieścić zadań w mniej niż podłoga z połowy z nich.

Algorytmy aproksymacyjne - SBP

Pomysł: sprowadzić problem stochastyczny do deterministycznego, przez zamianę zmiennej losowej na pewną stałą wartość.

Propozycja

$ES(X) = \frac{1}{N_X}$ gdzie N_X jest najmnejszą liczbą dla której $P(\sum_{i=1}^{N} U_i > 1) < \eta$ gdzie U_i są niezależne o rozkładzie takim samym jak X

Wyniki

- Rozkłady Poissona: Aproksymacja dla BP przechodzą bez zmian
- Rozkłady normalne: Aproksymacje przechodzą przy zwiększeniu pojemności maszyn o 49.6%

Dane: Google Trace 2011

Dane

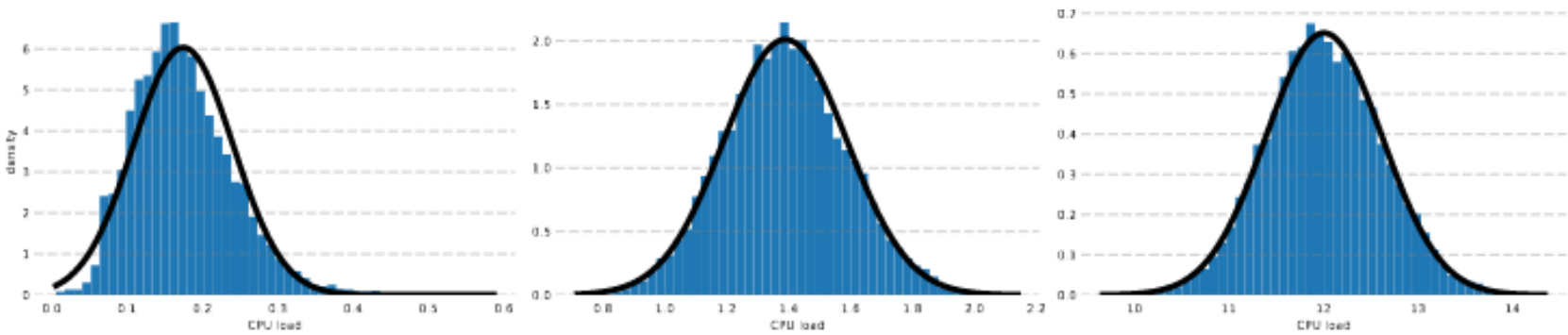
- 12600 zadań
- 30 dni
- średnie zużycie zasobów w przedziale 5 min
- chwilowe zużycie (średnia z 1sek) z losowych momentów w przedziałach 5 min

Uwaga

W 2019 Google opublikowało nowy trace z danymi z 8 klastrów (96.4k zadań) i dodatkowymi statystykami.

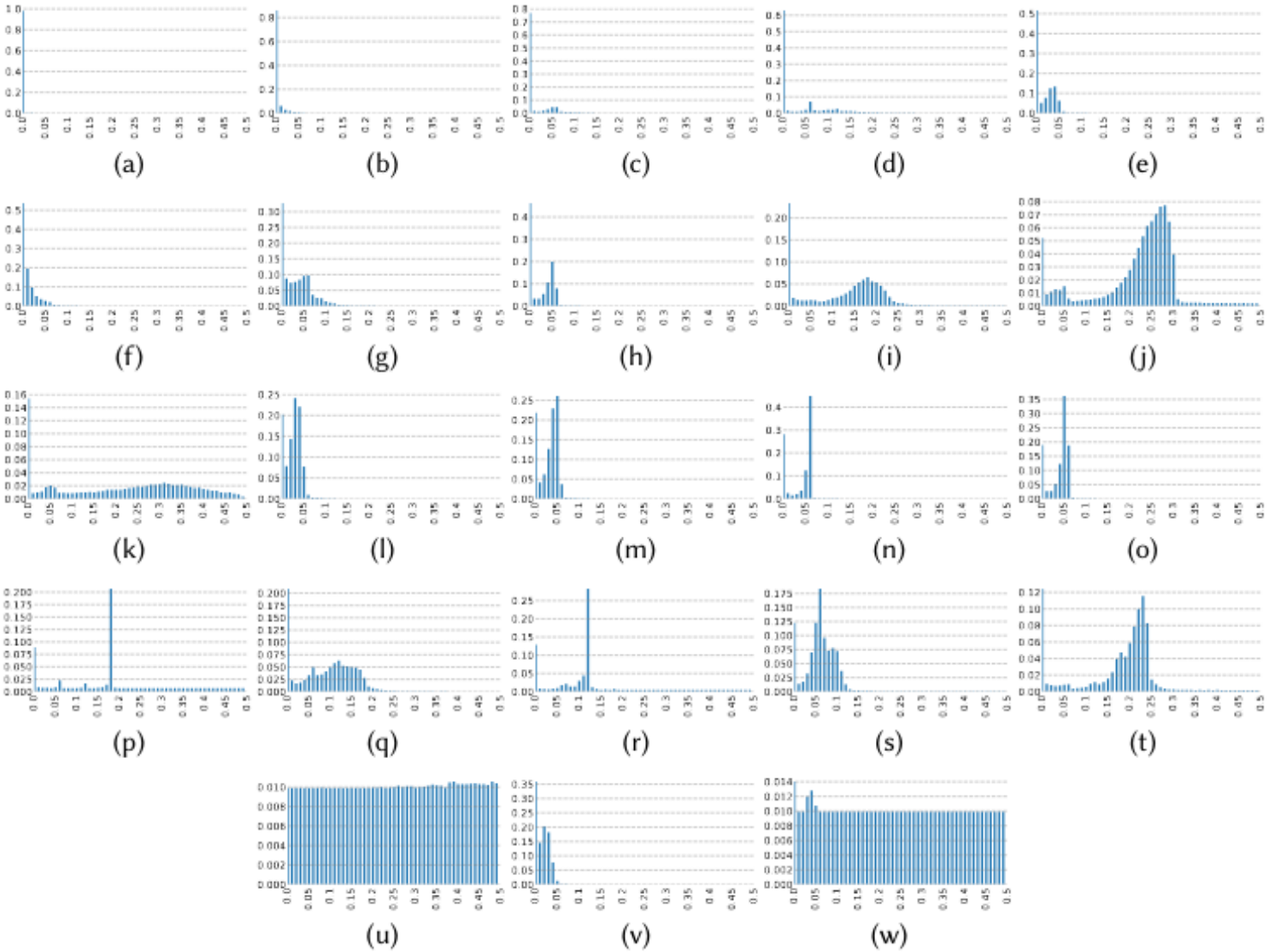
Wyniki negatywne

- Niezależność od czasu: nie zachodzi dla mniej więcej $\frac{1}{4} - \frac{1}{3}$ zadań.
- Zidentyfikowano ok. 23 istotnie różnych rozkładów
- Centralne Twierdzenie Graniczne (suma zmiennych losowych dąży do rozkładu normalnego): dla 50 losowych zadań prawdopodobieństwo, że suma NIE ma rozkładu normalnego to ok. 78%, dla 500 to ok. 33% (a średnie obciążenie to 12.5 największej maszyny)



(a) $N = 10$ (b) $N = 50$ (c) $N = 500$

Rozkłady



GPA: Gaussian Percentile Approximation

Wystarczy przewidzieć prawdopodobieństwo przepełnienia maszyny: $P(X > c) = 1 - CDF_X(c)$

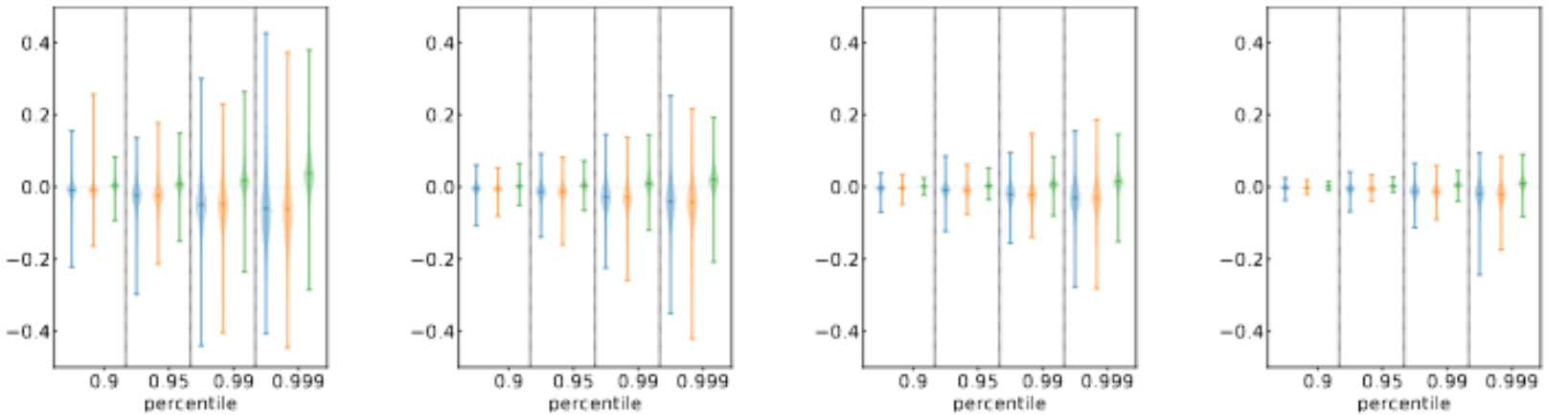
Metody dopasowania rozkładu normalnego $N(\mu, \sigma^2)$ do danych

- GPA-g: $\mu = \sum_i \mu_i,\ \sigma = \sqrt{\sum_i \sigma_i^2}$
- GPA-id:
  $X(t) = \sum_i x_i(t),\ \mu = \frac{1}{T} \sum_{t=1}^{T} X(t),\ \sigma = \sqrt{\frac{1}{T} \sum_{t=1}^{T} (X(t) - \mu)^2}$
- GPA-sqrt: $X(t) = \sqrt{\sum_i x_i(t)}$ —||—

GPA: Gaussian Percentile Approximation

Dopasowanie rozkładu

Aby otrzymać $p-ty$ percentyl dla $N(\mu, \sigma^2)$ obliczamy funkcję odwrotną do dystrybuanty w punkcie $p/100$: $F^{-1}_{\mu,\sigma}(p/100)$



(a) 10 tasks in a bin (b) 20 tasks in a bin (c) 30 tasks in a bin (d) 50 tasks in a bin

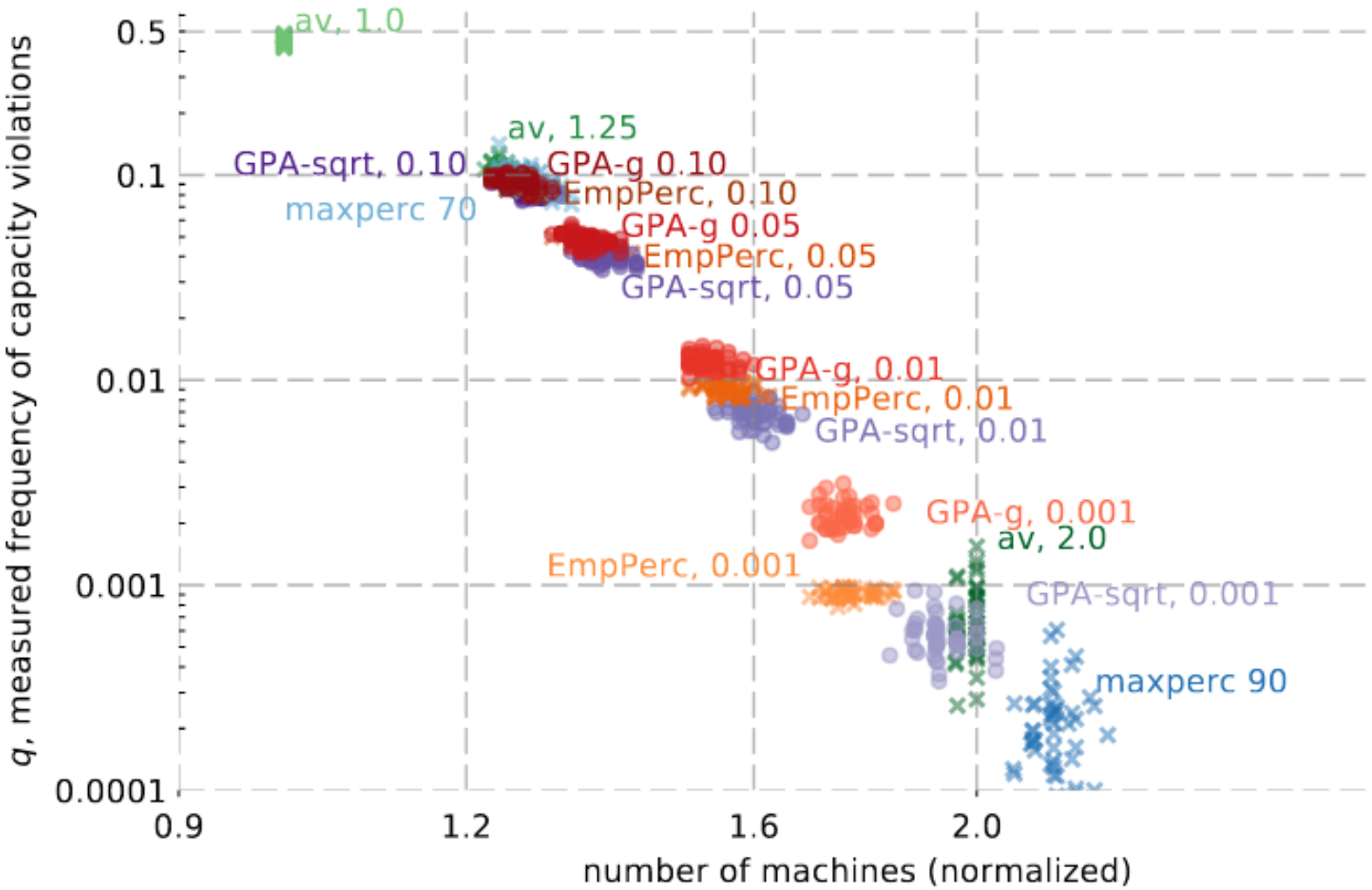
Przedziały i średnie wartości dla względnej różnicy między estymacją a danymi empirycznymi: $(F^{-1}_{\mu,\sigma}(k) - P_k)/P_k$. Kolejność: GPA-g, GPA-id, GPA-sqrt

Algorytmy

Best-fit gdzie warunek przepełnienia estymujemy za pomocą:

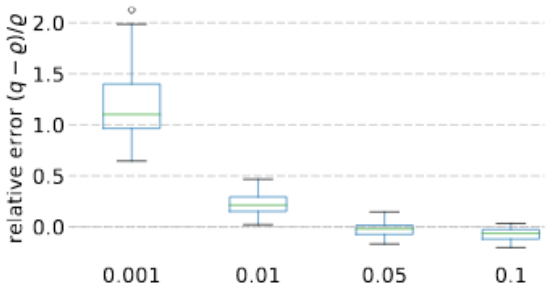
- GPA-g / GPA-sqrt
- EmpPerc: j-ty percentyl sumy obciążenia maszyny i danych empirycznych zadania
- EmpPercExtended: EmpPerc gdzie jako dane o zadaniu bierzemy losową próbkę z nich
- Cantelli: Nierówność Cantelliego: $P(X > \mu + b\sigma) < \frac{1}{1+b^2}$
- av: $f\mu$ gdzie f to stała $\in \{1.0, 1.25, 2.0\}$
- perc: percentyl danych zadania

Porównanie algorytmów

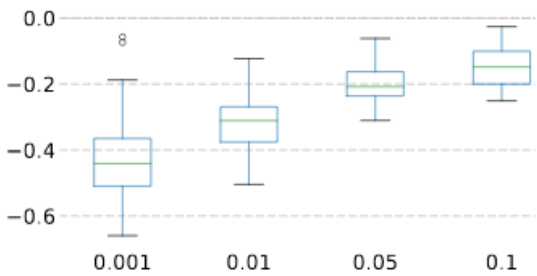


Wpływ SLO na wymagania i błędy

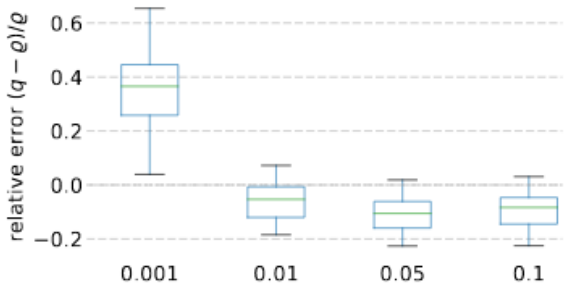
SLO	required bin size	bin size relative to 90% SLO
90%	1.294	1.00
99%	1.488	1.15
99.9%	1.643	1.27
99.99%	1.760	1.36



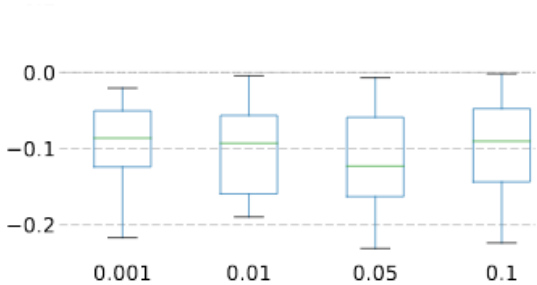
(a) *GPA-g*



(b) *GPA-sqrt*

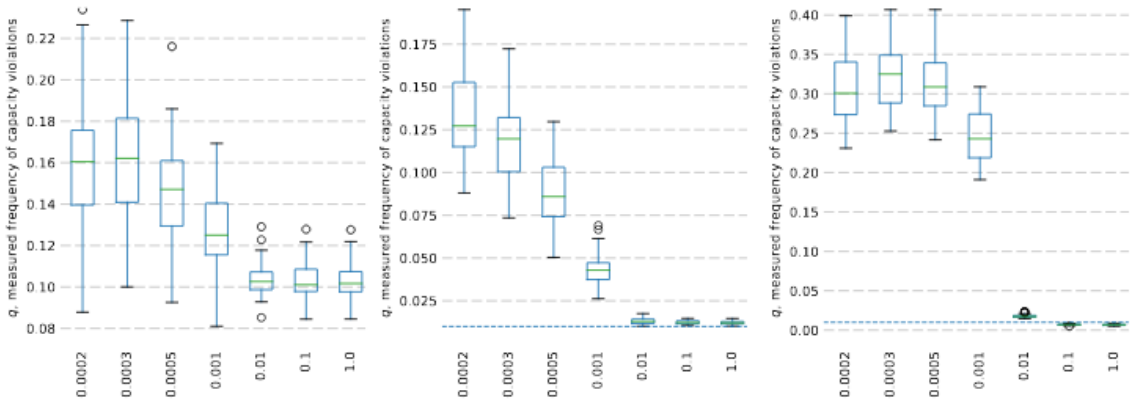


(c) *EmpPercExtended*

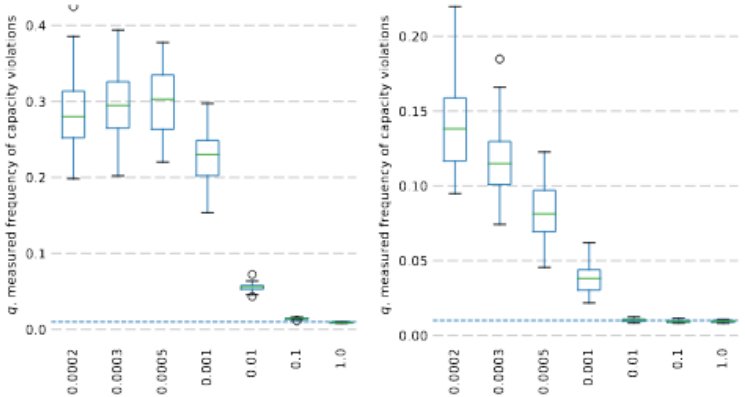


(d) *EmpPerc*

Wpływ liczby danych na błędy



(a) av 1.25 (b) *GPA-g* $\varrho = 0.01$ (c) *GPA-sqrt* $\varrho = 0.01$



(d) *EmpPerc* $\varrho = 0.01$ (e) *EmpPercExtended* $\varrho = 0.01$

Oś pozioma: ułamek danych za pomocą których algorytm podejmował decyzję (0.001 to 10 odczytów)
Oś pionowa: częstotliwość przekraczania limitów

Algorytmy aproksymacyjne - SBP

Pomysł: sprowadzić problem stochastyczny do deterministycznego, przez zamianę zmiennej losowej na pewną stałą wartość.

Propozycja

$ES(X) = \frac{1}{N_X}$ gdzie N_X jest najmnejszą liczbą dla której $P(\sum_{i=1}^{N} U_i > 1) < \eta$ gdzie U_i są niezależne o rozkładzie takim samym jak X

Wyniki

- Rozkłady Poissona: Aproksymacja dla BP przechodzą bez zmian
- Rozkłady normalne: Aproksymacje przechodzą przy zwiększeniu pojemności maszyn o 49.6%

Aproksymacja SBP dla rozkładu Poissona

$$P(Pois(\mu) = k) = \frac{\mu^k e^{-\mu}}{k!}, \quad Pois(\mu_1) + Pois(\mu_2) = Pois(\mu_1 + \mu_2)$$

Weźmy wartość μ_p spełniającą: $P(Pois(\mu_p) > 1) = \nu$.

Możemy sprowadzić problem stochastyczny do deterministycznego przez zastąpienie każdej zmiennej przez jej wartość oczekiwaną, a pojemność maszyn przez μ_p.

Aproksymacja SBP dla rozkładu normalnego

$$g_{N(\mu,\sigma^2)}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \; N(\mu_1,\sigma_1^2)+N(\mu_2,\sigma_2^2) = N(\mu_1+\mu_2,\sigma_1^2+\sigma_2^2)$$

Weźmy wartość Z_η, spełniającą $P(N(0,1) > Z_\eta) = \eta$
Wtedy $P(N(\mu,\sigma^2) > 1) = \eta \iff \mu + Z_\eta\sigma \leq 1$
Możemy zastąpić zmienną o rozkładzie $N(\mu,\sigma^2)$ przez $\frac{1}{N}$ gdzie N t,że $N\mu + Z_\eta\sqrt{N}\sigma = 1$ i rozwiązać deterministyczny problem z maszynami pojemności 1.
Można pokazać, że jeśli $\frac{1}{N_1} + \frac{1}{N_2} \leq 1$ to

$$\mu_1 + \mu_2 + Z_\eta\sqrt{\sigma_1^2 + \sigma_2^2} \leq 1.496$$

Aproksymacja SBP dla rozkładu normalnego V2

Teza

Algorytm Next-fit z przedmiotami posortowanymi malejąco po $\frac{\sigma^2}{\mu}$ (VMR: Variance To Mean Ratio) jest 2-aproksymacją gdy zmienne wejściowe mają rozkład normalny.

Rozpatrzmy następujący program matematyczny (PM) dla minimalnego m dla którego istnieje rozwiązanie

$$\sum_{i=1}^{n} x_{ij}\mu_i + Z_\eta \sqrt{\sum_{i=1}^{n} x_{ij}\sigma_i^2} \leq 1 \text{ dla } 1 \leq j \leq m$$

$$\sum_{j=1}^{m} x_{ij} = 1 \text{ dla } 1 \leq i \leq n$$

$$x_{ij} \leq 0 \text{ dla } 1 \leq i \leq n, 1 \leq j \leq m$$

Zmienne x_{ij} oznaczają część przedmiotu i w kubełku j.

Aproksymacja SBP dla rozkładu normalnego V2

Lemat

Istnieje rozwiązanie PM spełniające:

$$\forall_{k,l} \forall_{i<j} x_{kj} > 0 \wedge x_{li} > 0 \implies \frac{\sigma_l^2}{\mu_l} \leq \frac{\sigma_k^2}{\mu_k}$$

Dowód techniczny, szkic: weźmy rozwiązanie MP z maksymalnym leksykograficznie wektorem wariancji kubełków. Załóżmy, że teza dla niego nie zachodzi, tj.

$$_{k,l} i_{<j} x_{kj} > 0 \wedge x_{li} > 0 \wedge \frac{\sigma_l^2}{\mu_l} > \frac{\sigma_k^2}{\mu_k}$$

Weźmy $\delta = \frac{\min x_{kj}\mu_k x_{li}\mu_l}{\mu_l}$, zmniejszmy x_{li} o δ i zwiększmy x_{lj} o δ, zwiększmy x_{ki} o największe δ' które się zmieści, zmniejszmy x_{kj} o δ'

Aproksymacja SBP dla rozkładu normalnego V2

Lemat

Istnieje rozwiązanie PM spełniające:

$$\forall_{k,l} \forall_{i<j} x_{kj} > 0 \wedge x_{li} > 0 \implies \frac{\sigma_l^2}{\mu_l} \leq \frac{\sigma_k^2}{\mu_k}$$

Otrzymaliśmy inne rozwiązanie MP z większym leksykograficznie wektorem wariancji (techniczny rachunek) ↯

Wniosek

Ułamkowy Next-fit z przedmiotami posortowanymi malejąco po VMR znajduje rozwiązanie MP.

Aproksymacja SBP dla rozkładu normalnego V2

Weźmy pewną instancję problemu I, oraz rozwiązanie otrzymane za pomocą Next-fit przy przedmiotach posortowanych po MP, oznaczmy liczbę użytych kubełków przez B.

Weźmy prostszą instancję I': Bierzemy tylko przedmioty które trafiły do nieparzystych kubełków i pierwsze przedmioty które się do nich nie zmieściły pomniejszone tak aby się zmieściły.

Oznaczając przez FRAC algorytm rozwiązujący MP, OPT algorytm optymalny, mamy szacowanie:

$$\lceil B/2 \rceil = FRAC(I') \leq FRAC(I) \leq OPT(I)$$

Aproksymacja SBP online dla rozkładu normalnego

Algorytm

Weźmy $\epsilon > 0$ i zdefiniujmy klasy przedmiotów jako przedziały VMR:

$$[0, \epsilon^2),\ [\epsilon^2(1+\epsilon)^{k-1}, \epsilon^2(1+\epsilon)^k,\ [\epsilon^2(1+\epsilon)^k, 1]$$

Gdzie $C = \lceil \frac{8}{\epsilon} \ln(\frac{1}{\epsilon}) \rceil,\ k = 1, \ldots, C$
Umieszczamy nowy element w kubełku zawierającym tylko elementy z jego klasy zgodnie ze strategią First-fit.

Jest to algorytm online $2(1+\epsilon)$-aproksymający SBP dla rozkładu normalnego.

Dziękuję za uwagę!

Pytania?