# شبیه‌سازی صف

ابراهیم بیاگوی

# بهینه‌سازی‌ها

- انتخاب زبانی بهتر برای برنامه‌های بهینه
- استفاده نکردن از اختصاص حافظه پویا و مصرف کمتر حافظه
- استفاده از الگوریتم بهینه‌تر برای تولید اعداد شبه‌تصادفی
- بهینه‌سازی الگوریتم انتخاب (اسلاید بعد)
- استفاده از Release به جای Debug
- نتیجه برای ۱۰ میلیون:
  - از ۱۰ ثانیه کد اصلی، به ۰٫۳ ثانیه
  - مصرف گیگابایتی حافظه، به مصرف ۷۰۰کیلوبایتی

```
inline int fastrand() {
  g_seed = (214013 * g_seed + 2531011);
  return (g_seed >> 16) & 0x7FFF;
}
```

دو برابر سرعت نسبت به std::rand

- نگه‌داشتن تست‌ها جهت اطمینان از صحت
- دو پیاده‌سازی (هر دو انجام شد):
  - پیاده‌سازی بی‌نهایت با امکان ذخیره حالت
  - پیاده‌سازی با محدودیت uint64_t ولی با سرعت بیشتر

1.8446744e+19

محدودیت در uint64_t

# انتخاب از بین موارد

- بهبود الگوریتم انتخاب مورد

```
function* itemPicker(stream, items) {
    for (let value of stream)
        for (let item in items) {
            if (Math.round(value * 1000) / 1000 === 0) {
                yield +item;
                break;
            }
            value -= items[item];
            if (Math.round(value * 1000) / 1000 <= 0) {
                yield +item;
                break;
            }
        }
}
```

قبل

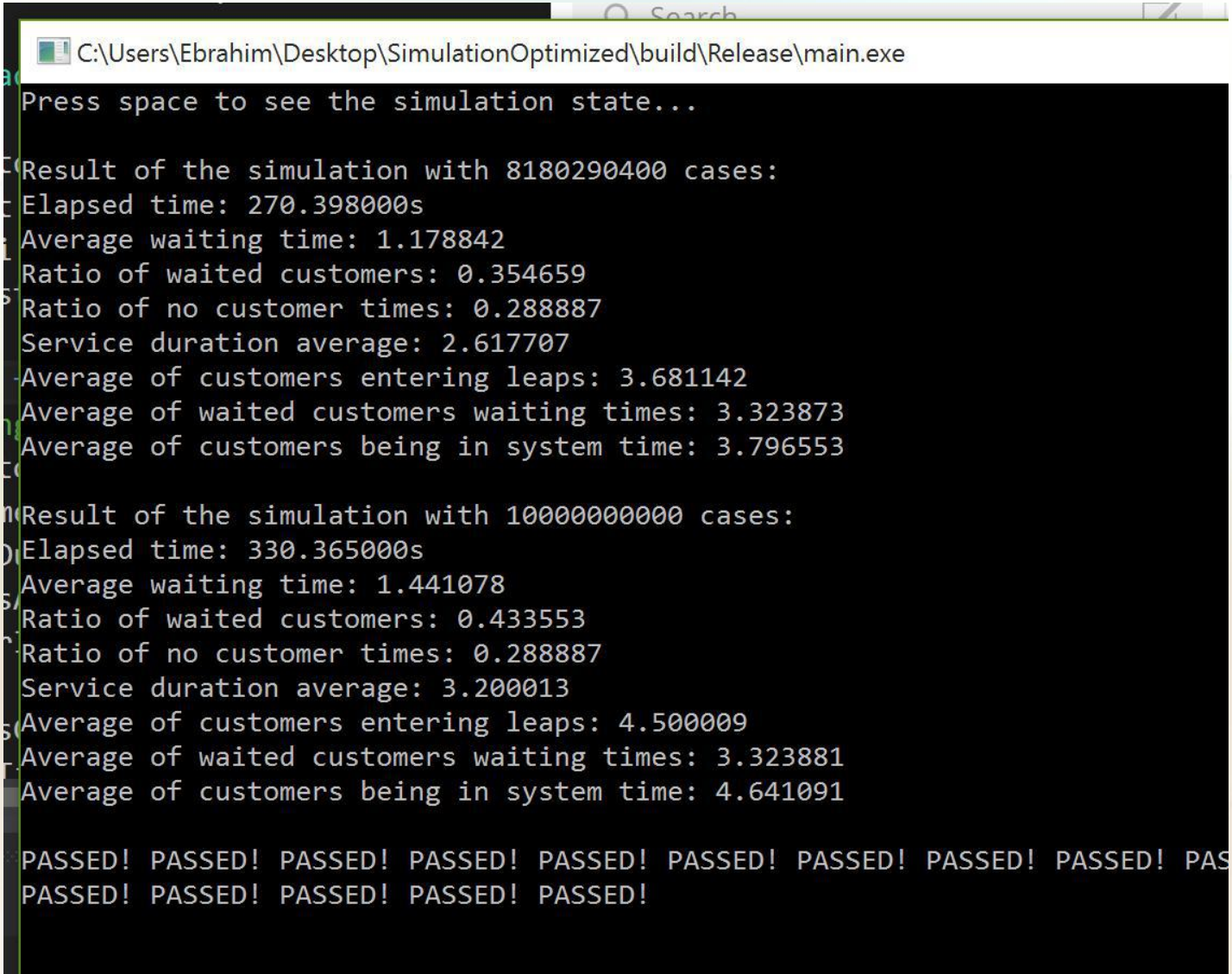
بعد

```
class ItemPicker {
  uint8_t* items;
  int* aggregatedPossbilities;
  unsigned long n;

public:

  ItemPicker(std::map<uint8_t, float> possibilities) {
    n = possibilities.size();
    items = new uint8_t[n];
    aggregatedPossbilities = new int[n];
    float aggreagation = 0;
    int i = 0;
    for (auto const item : possibilities) {
      aggreagation += item.second;
      aggregatedPossbilities[i] = aggreagation * RAND_MAX;
      items[i] = item.first;
      i++;
    }
  }

  inline uint8_t pick(int value) {
    for (int i = 0; i < n; ++i)
      if (value <= aggregatedPossbilities[i])
        return items[i];
  }
```

بهینه‌سازی: ورودی احتمال به صورت عدد صحیح

قابل بهینه‌سازی در صورت تعداد بالای گزینه‌ها برای انتخاب (bsearch)

# اجرای ۱۰ میلیارد مورد در ۳۳۰ ثانیه



# انجام در متلب

## استفاده از امکانات متلب و تاکید بر تولید و استفاده از آرایه/ماتریس‌ها

simulation.m

```
21 -  n = 100000000;
22
23 -  arrivals = randsample(arrivalDistribution(:, 1)', n, true, arrivalDistribution(:, 2)');
24 -  services = randsample(serviceDistribution(:, 1)', n, true, serviceDistribution(:, 2)');
25
26 -  fprintf('N: %d\n', n);
27    %%%%%%%%%%%%%%%%% QUEUE SIMULATION %%%%%%%%%%%%%%%%%
28 -  waitingTimeAvg = main(arrivals, services, n);
29    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
30
31 -  arrivalMean = mean(arrivals);
32 -  arrivalVar = var(arrivals) ^ .5;
33 -  servicesMean = mean(services);
34 -  servicesVar = var(services) ^ .5;
35
36 -  lambda = 1 / arrivalMean;
37 -  tau = servicesMean;
38 -  ro = lambda / (1 / tau);
39 -  ca = arrivalVar / arrivalMean;
40 -  cs = servicesVar / servicesMean;
41
42 -  fprintf('Expected: %f, Actual: %f\n', ...
43      ... % Kingman's formula (estimation)
44      (ro / (1 - ro)) * (((ca ^ 2) + (cs ^ 2)) / 2) * tau, ...
45      waitingTimeAvg);
```

$$\mathbb{E}(W_q) \approx \left(\frac{\rho}{1-\rho}\right)\left(\frac{c_a^2 + c_s^2}{2}\right)\tau$$

Command Window

New to MATLAB? See resources for Getting Started.

```
N: 100000000
Expected: 1.659791, Actual: 1.441394
```