Go + SMTP + RabbitMQ = PostmanQ или как мы рассылаем ~20K писем в минуту.

Соломонов Алексей, Actionpay

План

- Кто мы и что демаем?
- Зачем нам свой MTA?
- Немного о PostmanQ
- Архитектура
- Производительность
- На бою
- Итоги

AliExpress

М.видео

actionpay

cpa - сеть

СВЯЗНОЙ

ULMART.RU

110090 вебмастеров

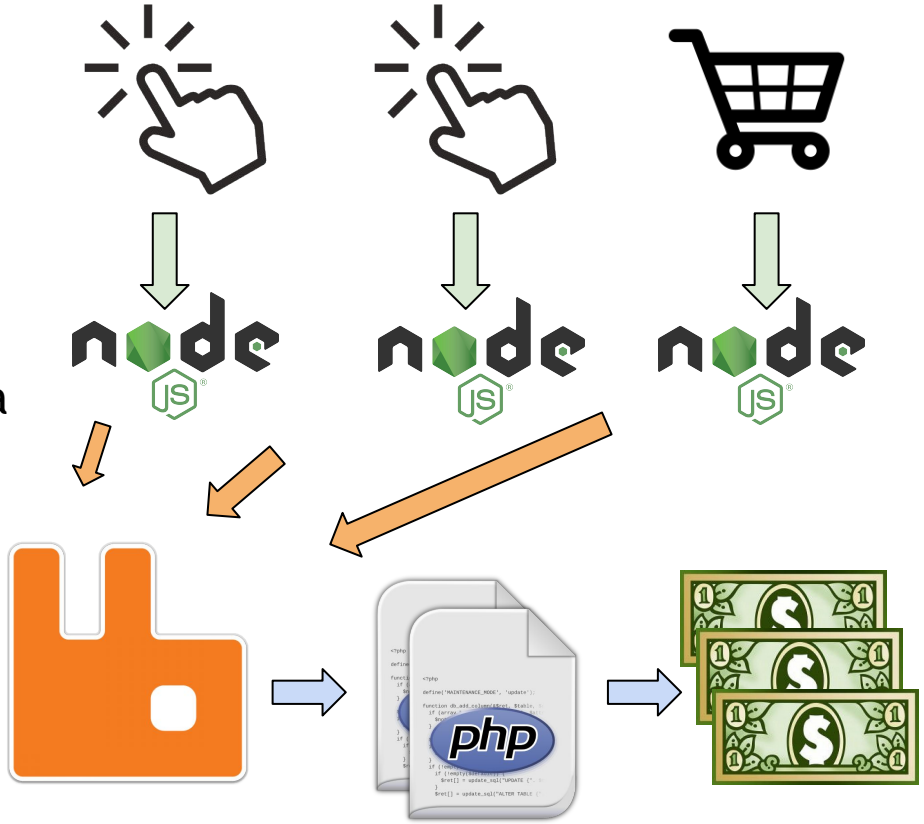
897 офферов

Что мы делаем?

Получаем клик от пользователя
Перенаправляем его в магазин
Получаем действие от магазина
Получаем подтверждение от магазина
Выплачиваем вознаграждение

~4 млн кликов в день

~ 150 тыс. действий в день

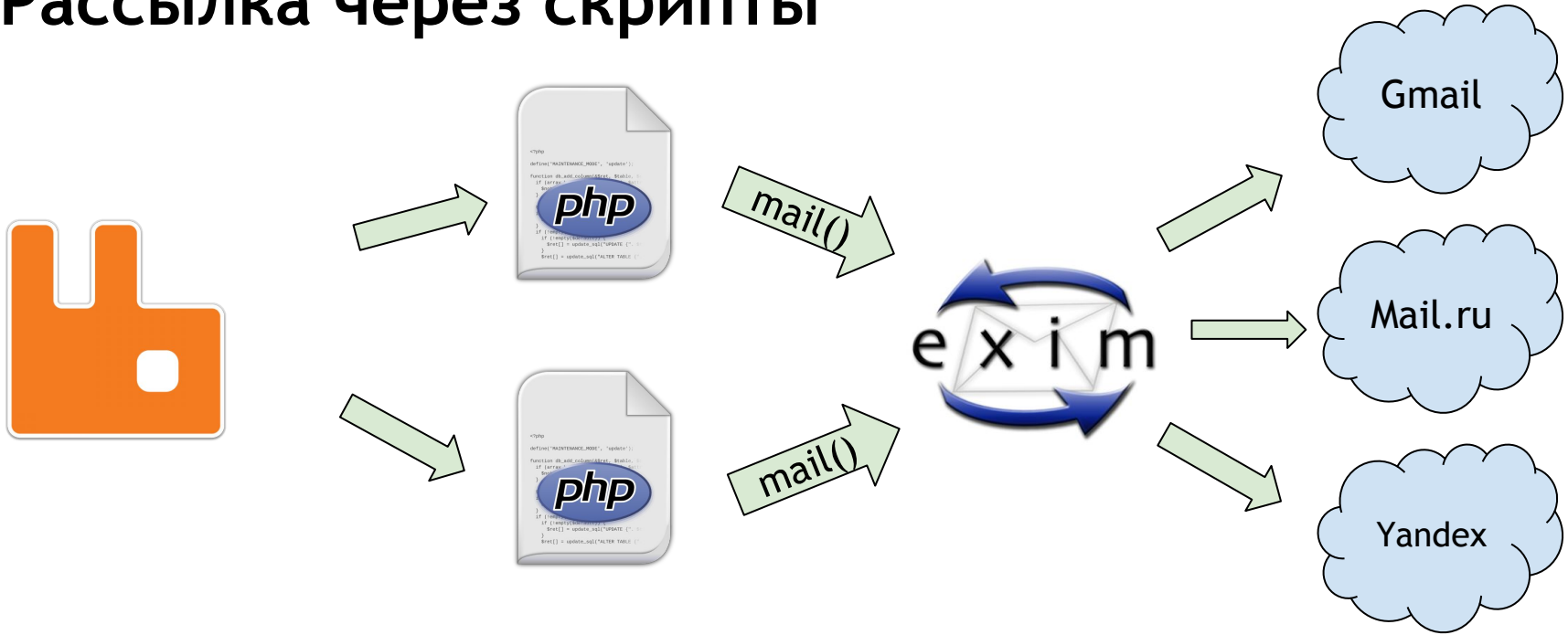


Откуда столько почты?

- Рассылки о новых офферах, акциях
- Изменение настроек оффера

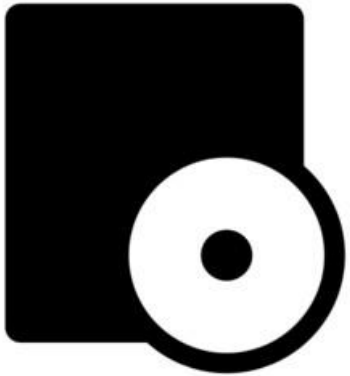
- Оповещения из личного кабинета

Рассылка через скрипты



Рассылка через скрипты

Рассылка через C++ решение



Gmail

Mail.ru

Yandex

- 100 писем в секунду
- Ограничение на кол-во IP
- Одна очередь
- Решение стало платным и ограничило скорость
- Падения при большом количестве писем

Рассылка через C++ решение

- 100 писем в секунд
- Ограничение на
- Одна очередь
- Решение стал ым
  и ограничило
- Падения при

Слишком много
НО!

Mail.ru

Yandex

Требования

- Высокая производительность
- TLS
- DKIM
- Простота конфигурации
- Стабильная работа

Требования к ЯП

- Производительный
- Разумно потребляющий ресурсы
- Удобная модель многопоточности, возможно параллелизм
- Хороший набор сетевых библиотек
- Простота разработки
- Владение языком другими членами команды

Почему Go?

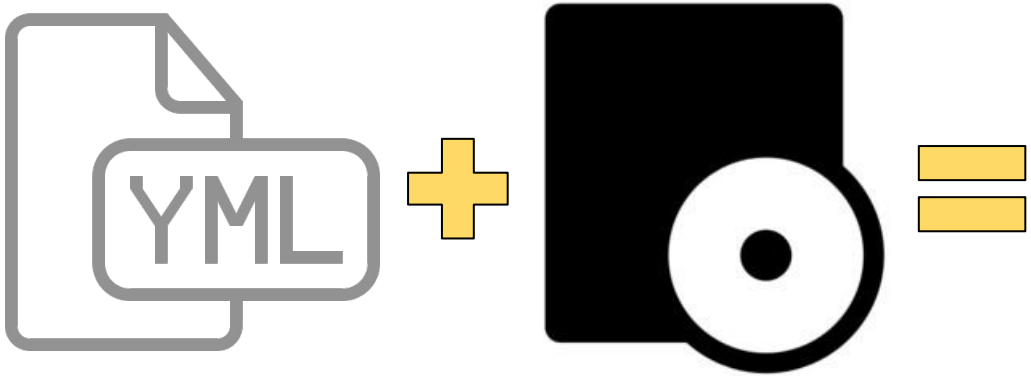
- Производительность сопоставима с java
- Адекватное потребление памяти:
  - простое веб приложение ~40MB при 38K rps
  - PostmanQ ~840MB при ~20K писем в мин
- Параллелизм из коробки
- Стандартная библиотека(net, net/smtp, crypto/tls)
- Очень простой синтаксис

Возможности PostmanQ

- Работа с несколькими AMQP и очередями
- Рассылка ведется с одного домена, но разных IP
- Использование TLS
- Подписка DKIM у писем
- Исключение из рассылки писем по домену
- Повторная отправка писем
- Распределение не отправленных писем
- Логирование в консоль или файл

PostmanQ + утилиты:

- pmq-grep - поиск логов по получателю
- pmq-report - просмотр ошибок
- pmq-publish - перекладывает письма между очередями



- 1 мастер процесс
- жизненный цикл:
  - инициализация
  - запуск
  - остановка
- логика разбита по пакетам
- всё делают goroutines

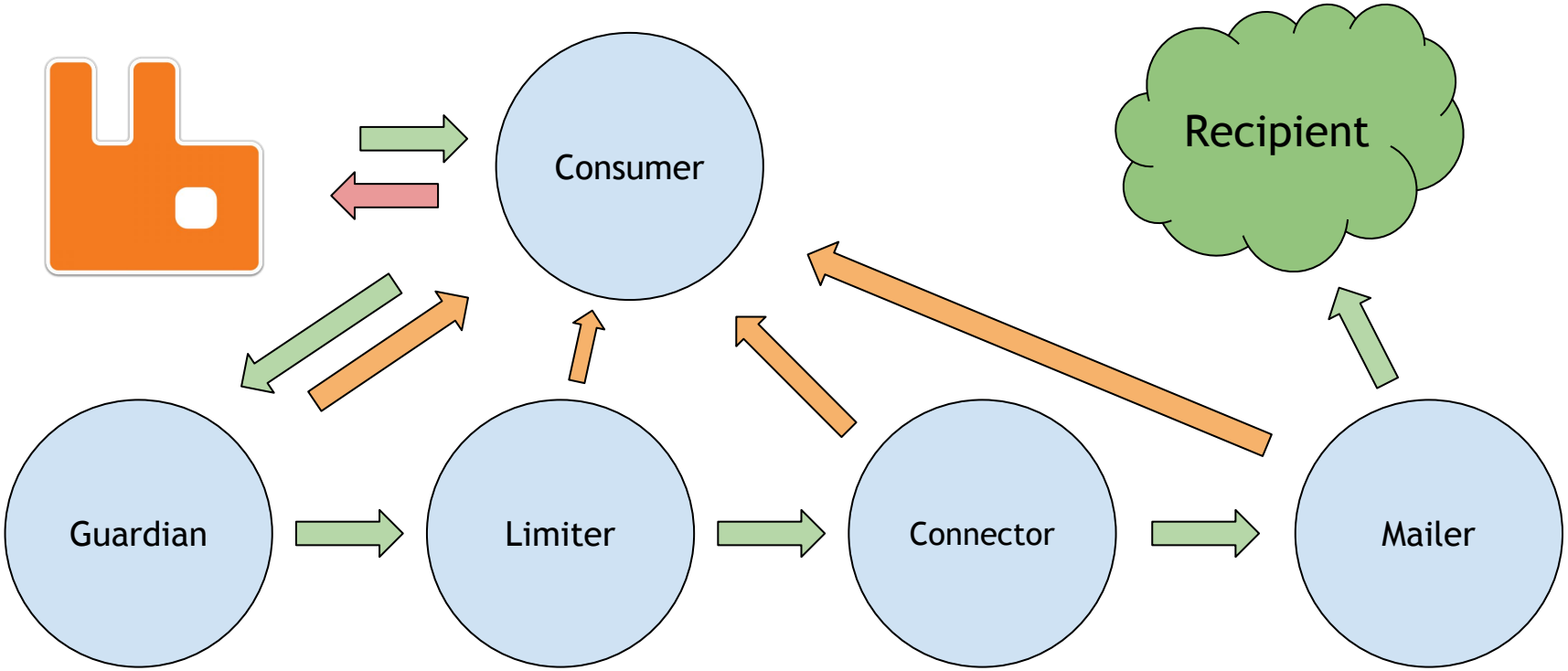
PostmanQ создает очереди в RabbitMQ:

postmanq	**outbox**	D	idle	0	17	17	0.00/s	0.00/s	0.00/s
postmanq	**outbox.dlx.day**	D TTL DLX	idle	0	0	0			
postmanq	**outbox.dlx.fifty.minutes**	D TTL DLX	idle	6	0	6	0.00/s		
postmanq	**outbox.dlx.five.minutes**	D TTL DLX	idle	0	0	0	0.00/s		
postmanq	**outbox.dlx.forty.minutes**	D TTL DLX	idle	83	0	83	0.00/s		
postmanq	**outbox.dlx.hour**	D TTL DLX	idle	1	0	1	0.00/s		
postmanq	**outbox.dlx.twenty.minutes**	D TTL DLX	idle	0	0	0	0.00/s		
postmanq	**outbox.failure.connection**	D	idle	240	0	240	0.00/s	0.00/s	
postmanq	**outbox.failure.recipient**	D	idle	9,123	0	9,123	0.00/s	0.00/s	
postmanq	**outbox.failure.technical**	D	idle	2,109	0	2,109	0.00/s	0.00/s	
postmanq	**outbox.failure.unknown**	D	idle	8,911	0	8,911	0.00/s	0.00/s	
postmanq	**outbox.not.send**	D	idle	1,185	0	1,185	0.00/s	0.00/s	0.00/s

Сообщение из RabbitMQ:

```
{
    "envelope": "sender@mail.foo",
    "recipient": "recipient@mail.foo",
    "body": "письмо с заголовками и содержимым"
}
```

Отправка письма:



Обработка пакетом своей части отправки:

```go
type Service struct {}

func (s *Service) OnRun() {
	for i := 0; i < s.WorkersCount; i++ {
		go newWorker(i + 1)
	}
}

func (s *Service) Events() chan *Event {
	return events
}
```

```go
func newWorker(id int) {
	worker := &Worker{id}
	worker.run()
}
func (w *Worker) run() {
	for event := range events {
		w.work(event)
	}
}
func (w *Worker) work(event *Event) {
	// do something
	event.Iterator.Next().Events() <- event
}
```

Механизмы многопоточности

- Каналы для взаимодействия
- Мьютексы для инициализации и проверки
- Засыпание goroutine на медленных операциях

Мьютексы

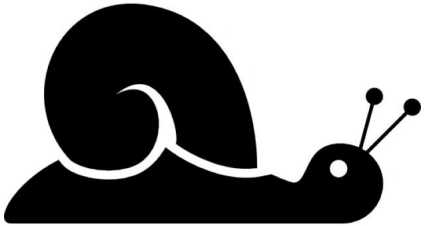
```
seekerMutex.Lock()
if _, ok := mailServers[hostnameTo]; !ok {
	mailServers[hostnameTo] = &MailServer{
		status:      LookupMailServerStatus,
		connectorId: event.connectorId,
	}
}
seekerMutex.Unlock()
mailServer := mailServers[hostnameTo]
```

Только для быстрых операции!

Засыпание goroutine

```
        goto connectToMailServer
connectToMailServer:
        //do something
        server := <-connectionEvent.servers
        switch server.status {
        case LookupMailServerStatus: goto waitLookup
        case SuccessMailServerStatus:
                connectionEvent.server = server
                connectorEvents <- connectionEvent
        case ErrorMailServerStatus: //do something
        }
        return
```

```
waitLookup:
        time.Sleep(common.App.Timeout().Sleep)
        goto connectToMailServer
        return
```



Для медленных операции...

Что влияет на производительность?

- Отклик получателя

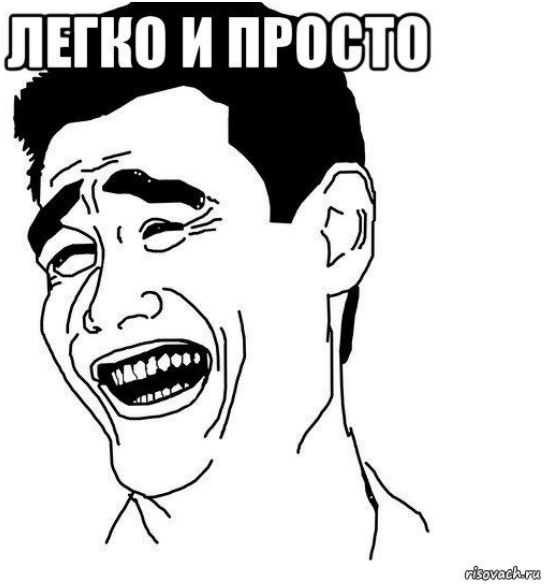
- Сетевые настройки
  - nofile
  - net.netfilter.nf_conntrack_max=1048576

- Количество IP
  - можно добавить

- PostmanQ
  - профилирование

Профилирование в Go

- Просто
- Наглядно
- Информативно
- github.com/pkg/profile для удобства



Как использовать?

```
import (
	// packages
	"github.com/pkg/profile"
)
func main() {
	defer profile.Start( profile.CPUProfile).Stop()
	// do something
}
```

```
./postmanq -f config.yaml
go tool pprof ./postmanq ./cpu.pprof
```

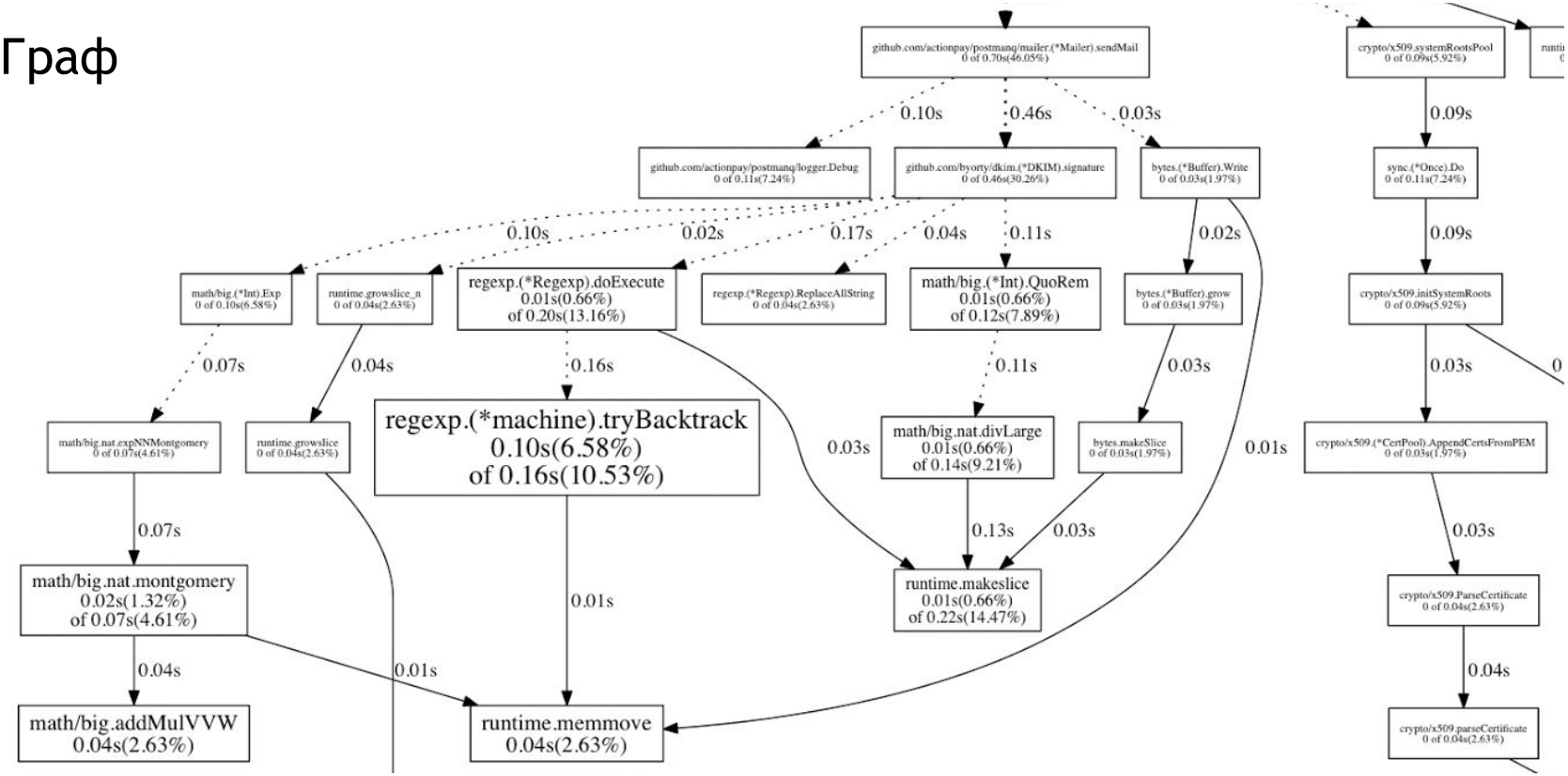
Топ вызовов

```
(pprof) top100
1520ms of 1520ms total (  100%)
Showing top 100 nodes out of 306 (cum >= 60ms)
      flat  flat%   sum%        cum   cum%
     190ms 12.50% 12.50%      190ms 12.50%  runtime.mach_semaphore_signal
     160ms 10.53% 23.03%      190ms 12.50%  syscall.Syscall
     110ms  7.24% 30.26%      110ms  7.24%  runtime.usleep
     100ms  6.58% 36.84%      160ms 10.53%  regexp.(*machine).tryBacktrack
      70ms  4.61% 41.45%       70ms  4.61%  runtime.cgocall
      70ms  4.61% 46.05%       70ms  4.61%  runtime.mach_semaphore_timedwait
      60ms  3.95% 50.00%       90ms  5.92%  runtime.(*mcentral).grow
      60ms  3.95% 53.95%       60ms  3.95%  runtime.mach_semaphore_wait
      60ms  3.95% 57.89%       60ms  3.95%  runtime.memclr
      40ms  2.63% 60.53%       40ms  2.63%  math/big.addMulVVW
      40ms  2.63% 63.16%      300ms 19.74%  runtime.mallocgc
      40ms  2.63% 65.79%       40ms  2.63%  runtime.memmove
      40ms  2.63% 68.42%       40ms  2.63%  runtime/internal/atomic.Cas
      30ms  1.97% 70.39%       70ms  4.61%  runtime.gentraceback
      30ms  1.97% 72.37%       30ms  1.97%  runtime.step
      30ms  1.97% 74.34%       30ms  1.97%  runtime.stkbucket
      20ms  1.32% 75.66%       20ms  1.32%  crypto/tls.(*halfConn).decrypt
      20ms  1.32% 76.97%       70ms  4.61%  math/big.nat.montgomery
      20ms  1.32% 78.29%       20ms  1.32%  runtime.kevent
      10ms  0.66% 78.95%       10ms  0.66%  crypto/elliptic.p256Sqr
```

Дерево вызовов

```
(pprof) tree
1520ms of 1520ms total (  100%)
Showing top 80 nodes out of 306 (cum >= 160ms)
----------------------------------------------------------+-------------
      flat  flat%   sum%        cum   cum%   calls calls% + context
----------------------------------------------------------+-------------
                                                  60ms 80.53% |   runtime.mallocgc
                                               14.50ms 19.47% |   runtime.notesleep
     190ms 12.50% 12.50%      190ms 12.50%                   | runtime.mach_semaphore_signal
----------------------------------------------------------+-------------
                                                  50ms 60.00% |   crypto/tls.(*Conn).Read
                                                  20ms 24.00% |   crypto/tls.(*Conn).Write
                                               13.33ms 16.00% |   crypto/tls.(*Conn).Close
     160ms 10.53% 23.03%      190ms 12.50%                   | syscall.Syscall
                                                  10ms   100% |   runtime.exitsyscall
----------------------------------------------------------+-------------
                                                1.07ms 80.53% |   runtime.mallocgc
                                                0.26ms 19.47% |   runtime.notesleep
     110ms  7.24% 30.26%      110ms  7.24%                   | runtime.usleep
----------------------------------------------------------+-------------
                                                 160ms   100% |   regexp.(*Regexp).doExecute
     100ms  6.58% 36.84%      160ms 10.53%                   | regexp.(*machine).tryBacktrack
```

Граф


github.com/actionpay/postmanq/mailer.(*Mailer).sendMail
0 of 0.70s(46.05%)
0.10s
0.46s
0.03s
crypto/x509.systemRootsPool
0 of 0.09s(5.92%)
0.09s
github.com/actionpay/postmanq/logger.Debug
0 of 0.11s(7.24%)
github.com/byorty/dkim.(*DKIM).signature
0 of 0.46s(30.26%)
bytes.(*Buffer).Write
0 of 0.03s(1.97%)
sync.(*Once).Do
0 of 0.11s(7.24%)
0.10s
0.02s
0.17s
0.04s
0.11s
0.02s
0.09s
math/big.(*Int).Exp
0 of 0.10s(6.58%)
runtime.growslice_n
0 of 0.04s(2.63%)
regexp.(*Regexp).doExecute
0.01s(0.66%)
of 0.20s(13.16%)
regexp.(*Regexp).ReplaceAllString
0 of 0.04s(2.63%)
math/big.(*Int).QuoRem
0.01s(0.66%)
of 0.12s(7.89%)
bytes.(*Buffer).grow
0 of 0.03s(1.97%)
crypto/x509.initSystemRoots
0 of 0.09s(5.92%)
0.07s
0.04s
0.16s
0.11s
0.03s
0.03s
math/big.nat.expNNMontgomery
0 of 0.07s(4.61%)
runtime.growslice
0 of 0.04s(2.63%)
regexp.(*machine).tryBacktrack
0.10s(6.58%)
of 0.16s(10.53%)
0.03s
math/big.nat.divLarge
0.01s(0.66%)
of 0.14s(9.21%)
bytes.makeSlice
0 of 0.03s(1.97%)
0.01s
crypto/x509.(*CertPool).AppendCertsFromPEM
0 of 0.03s(1.97%)
0.07s
0.13s
0.03s
0.03s
math/big.nat.montgomery
0.02s(1.32%)
of 0.07s(4.61%)
0.01s
runtime.makeslice
0.01s(0.66%)
of 0.22s(14.47%)
crypto/x509.ParseCertificate
0 of 0.04s(2.63%)
0.04s
0.01s
0.04s
math/big.addMulVVW
0.04s(2.63%)
runtime.memmove
0.04s(2.63%)
crypto/x509.parseCertificate
0 of 0.04s(2.63%)

Профилирование, итоги

- Мы инициализируем все, что можно при старте приложения
- Элементы коллекций сами удаляются/меняют состояние по таймауту
- Мы реализовали равномерное использование IP адресов через деление по модулю, вместо math/rand

Профил итоги

x2

- Мы инициализируе

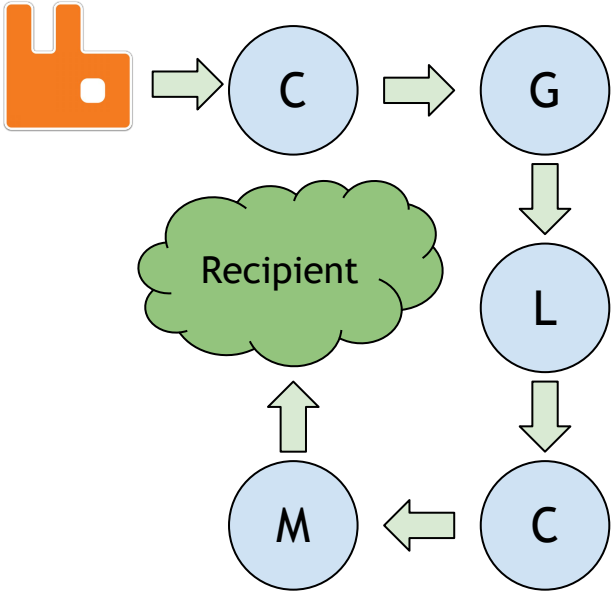
ллекци

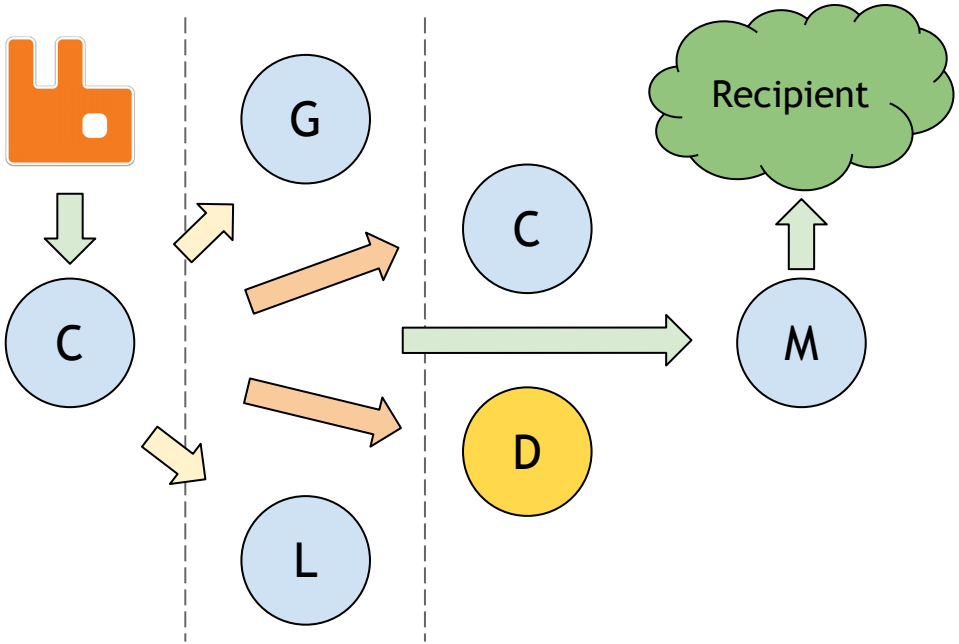
али равно з
модулю, вместо n/fa

Чего мы хотим?

- Избавиться от кастингов
- Избавиться от указателей
- Сделать событийную модель отправки

Событийная модель





На бою

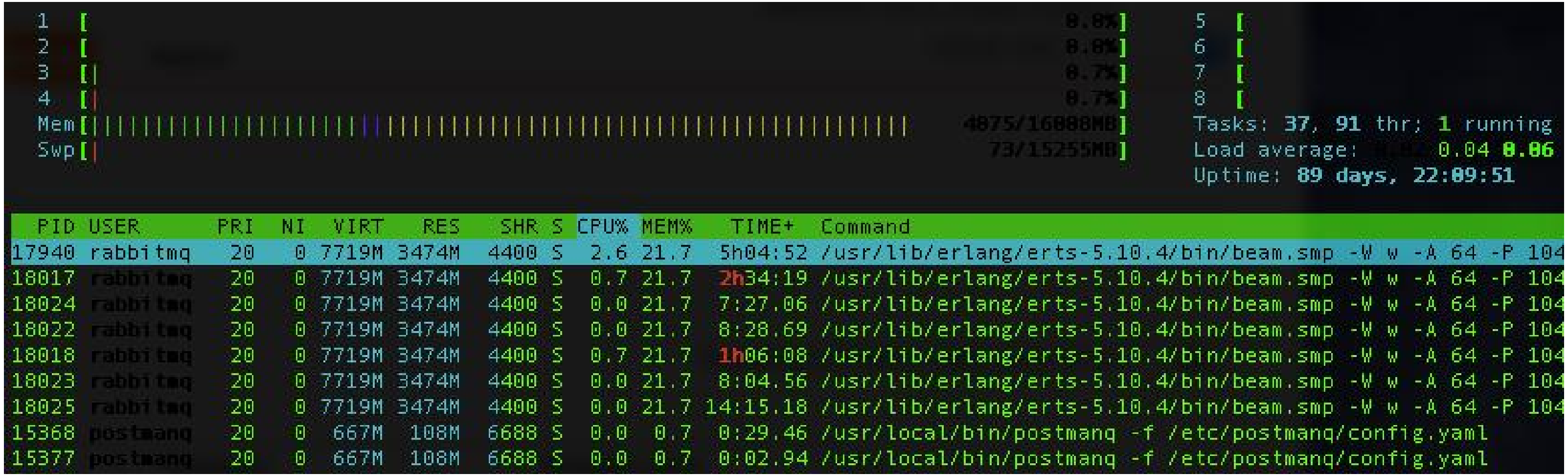
- Каждый проект использует свой PostmanQ
- Падения и утечки памяти не замечены
- Падения RabbitMQ при нехватке места на диске
- Мониторим отправку с помощью zabbix

Сервер рассылки

- Intel Xeon E3-1230, 8 ядер, hyper-threading
- 16Gb RAM
- 6 IP адресов
- ~310 - 340 писем в секунду, ~20K писем в минуту

Рассылка десятка писем:

- CPU ~0.2%
- Load average 0.04 0.06
- RAM 108Mb



```
  1  [                                                                 0.0%]     5  [
  2  [                                                                 0.0%]     6  [
  3  [|                                                                0.7%]     7  [
  4  [|                                                                0.7%]     8  [
  Mem[||||||||||||||||||||||||||||||||||||||||||||||||||||||||    4875/16088MB]     Tasks: 37, 91 thr; 1 running
  Swp[|                                                            73/15255MB]     Load average: [illegible] 0.04 0.06
                                                                                   Uptime: 89 days, 22:09:51

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
17940 rabbitmq   20   0 7719M 3474M  4400 S  2.6 21.7  5h04:52 /usr/lib/erlang/erts-5.10.4/bin/beam.smp -W w -A 64 -P 104
18017 rabbitmq   20   0 7719M 3474M  4400 S  0.7 21.7  2h34:19 /usr/lib/erlang/erts-5.10.4/bin/beam.smp -W w -A 64 -P 104
18024 rabbitmq   20   0 7719M 3474M  4400 S  0.0 21.7  7:27.06 /usr/lib/erlang/erts-5.10.4/bin/beam.smp -W w -A 64 -P 104
18022 rabbitmq   20   0 7719M 3474M  4400 S  0.0 21.7  8:28.69 /usr/lib/erlang/erts-5.10.4/bin/beam.smp -W w -A 64 -P 104
18018 rabbitmq   20   0 7719M 3474M  4400 S  0.7 21.7  1h06:08 /usr/lib/erlang/erts-5.10.4/bin/beam.smp -W w -A 64 -P 104
18023 rabbitmq   20   0 7719M 3474M  4400 S  0.0 21.7  8:04.56 /usr/lib/erlang/erts-5.10.4/bin/beam.smp -W w -A 64 -P 104
18025 rabbitmq   20   0 7719M 3474M  4400 S  0.0 21.7 14:15.18 /usr/lib/erlang/erts-5.10.4/bin/beam.smp -W w -A 64 -P 104
15368 postmanq   20   0  667M  108M  6688 S  0.0  0.7  0:29.46 /usr/local/bin/postmanq -f /etc/postmanq/config.yaml
15377 postmanq   20   0  667M  108M  6688 S  0.0  0.7  0:02.94 /usr/local/bin/postmanq -f /etc/postmanq/config.yaml
```

Рассылка сотен писем:

- CPU 52.3%, ~6,5% на ядро
- RAM 839Mb
- Load average 2.15 1.12

```
  1  [||                                                                                                  1.4%]     5  [||||||||
  2  [||||||||                                                                                            7.3%]     6  [||||||||||||
  3  [||||||||||                                                                                          18.5%]     7  [||
  4  [||||||||||                                                                                          12.8%]     8  [||||
  Mem[||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||4951/16088MB]     Tasks: 38, 94 thr; 1 running
  Swp[|                                                                                          67/15255MB]     Load average:  2.15 1.12
                                                                                                                 Uptime: 89 days, 23:58:18

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
30321 postmanq   20   0 1413M  839M  6680 S 52.3  5.2  6:01.44 /usr/local/bin/postmanq -f /etc/postmanq/config.yaml
17940 rabbitmq   20   0 7873M 3616M  4400 S 14.1 22.6  5h08:17 /usr/lib/erlang/erts-5.10.4/bin/beam.smp -W w -A 64 -P 104
30326 postmanq   20   0 1413M  839M  6680 S  8.7  5.2  0:20.91 /usr/local/bin/postmanq -f /etc/postmanq/config.yaml
30329 postmanq   20   0 1413M  839M  6680 S  8.0  5.2  0:34.16 /usr/local/bin/postmanq -f /etc/postmanq/config.yaml
30348 postmanq   20   0 1413M  839M  6680 S  8.0  5.2  0:08.23 /usr/local/bin/postmanq -f /etc/postmanq/config.yaml
30335 postmanq   20   0 1413M  839M  6680 S  6.7  5.2  0:32.64 /usr/local/bin/postmanq -f /etc/postmanq/config.yaml
30334 postmanq   20   0 1413M  839M  6680 S  6.0  5.2  0:34.02 /usr/local/bin/postmanq -f /etc/postmanq/config.yaml
30327 postmanq   20   0 1413M  839M  6680 S  6.0  5.2  0:32.92 /usr/local/bin/postmanq -f /etc/postmanq/config.yaml
18022 rabbitmq   20   0 7873M 3616M  4400 S  4.7 22.6  8:34.98 /usr/lib/erlang/erts-5.10.4/bin/beam.smp -W w -A 64 -P 104
```

Рассылка тысяч писем:

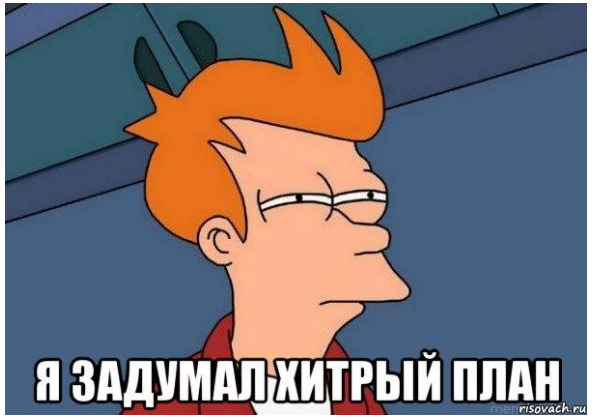
- CPU 764%, ~95% на ядро
- RAM 839Mb
- Load average 2.24 1.13

```
  1  [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||99.3%]     5  [||||||||||||||||||||||||||
  2  [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||99.3%]     6  [||||||||||||||||||||||||||
  3  [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||99.3%]     7  [||||||||||||||||||||||||||
  4  [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||100.0%]     8  [||||||||||||||||||||||||||
  Mem[||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||5063/16088MB]     Tasks: 38, 94 thr; 9 running
  Swp[|                                                                          67/15255MB]     Load average: 2.24 1.13
                                                                                                 Uptime: 89 days, 23:58:02

   PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 30321 postmanq   20   0 1413M  839M  6680 R 764.  5.2  5:26.00 /usr/local/bin/postmanq -f /etc/postmanq/config.yaml
 30329 postmanq   20   0 1413M  839M  6680 R 100.  5.2  0:29.94 /usr/local/bin/postmanq -f /etc/postmanq/config.yaml
 30332 postmanq   20   0 1413M  839M  6680 R 95.2  5.2  0:34.52 /usr/local/bin/postmanq -f /etc/postmanq/config.yaml
 30326 postmanq   20   0 1413M  839M  6680 R 95.2  5.2  0:18.71 /usr/local/bin/postmanq -f /etc/postmanq/config.yaml
 30331 postmanq   20   0 1413M  839M  6680 R 93.2  5.2  0:38.45 /usr/local/bin/postmanq -f /etc/postmanq/config.yaml
 30355 postmanq   20   0 1413M  839M  6680 R 92.5  5.2  0:08.34 /usr/local/bin/postmanq -f /etc/postmanq/config.yaml
 30327 postmanq   20   0 1413M  839M  6680 S 82.5  5.2  0:29.92 /usr/local/bin/postmanq -f /etc/postmanq/config.yaml
 30357 postmanq   20   0 1413M  839M  6680 R 78.5  5.2  0:09.83 /usr/local/bin/postmanq -f /etc/postmanq/config.yaml
 17940 rabbitmq   20   0 7991M 3722M  4400 S 37.3 23.3  5h08:13 /usr/lib/erlang/erts-5.10.4/bin/beam.smp -W w -A 64 -P 104
```

Планы

- Один сервер рассылки для всех проектов
- Написание полноценного SMTP-сервера
- Кластер?



Итоги

- PostmanQ удовлетворяет наши потребности
- Для нас это решение на вырост
- Мы получили инструмент, который стабильно работает
- Это первый опыт внедрения go, мы довольны, поэтому будем внедрять его в других проектах

Спасибо!

https://github.com/actionpay/postmanq

asolomonoff@gmail.com