

**PROFILIAVIMO VAIDMUO
ATLIEKANT ARCHITEKTŪROS
PAKEITIMUS**

JUSTAS BUTKUS

2014-05-08

Kas aš toks?

Time.ly Network Inc. | Lead Developer

- Viena didžiausių pasaulyje renginių DB
- Klientai: valstijos, renginių organizatoriai, naujienų portalai
- Sustainable, baigiamas pirmas VC etapas
- API užklausų apimtys -- 10.000 trx/s (15 ops avg.)

Anksčiau: mokėjimai.lt, eturas.

tl;dr

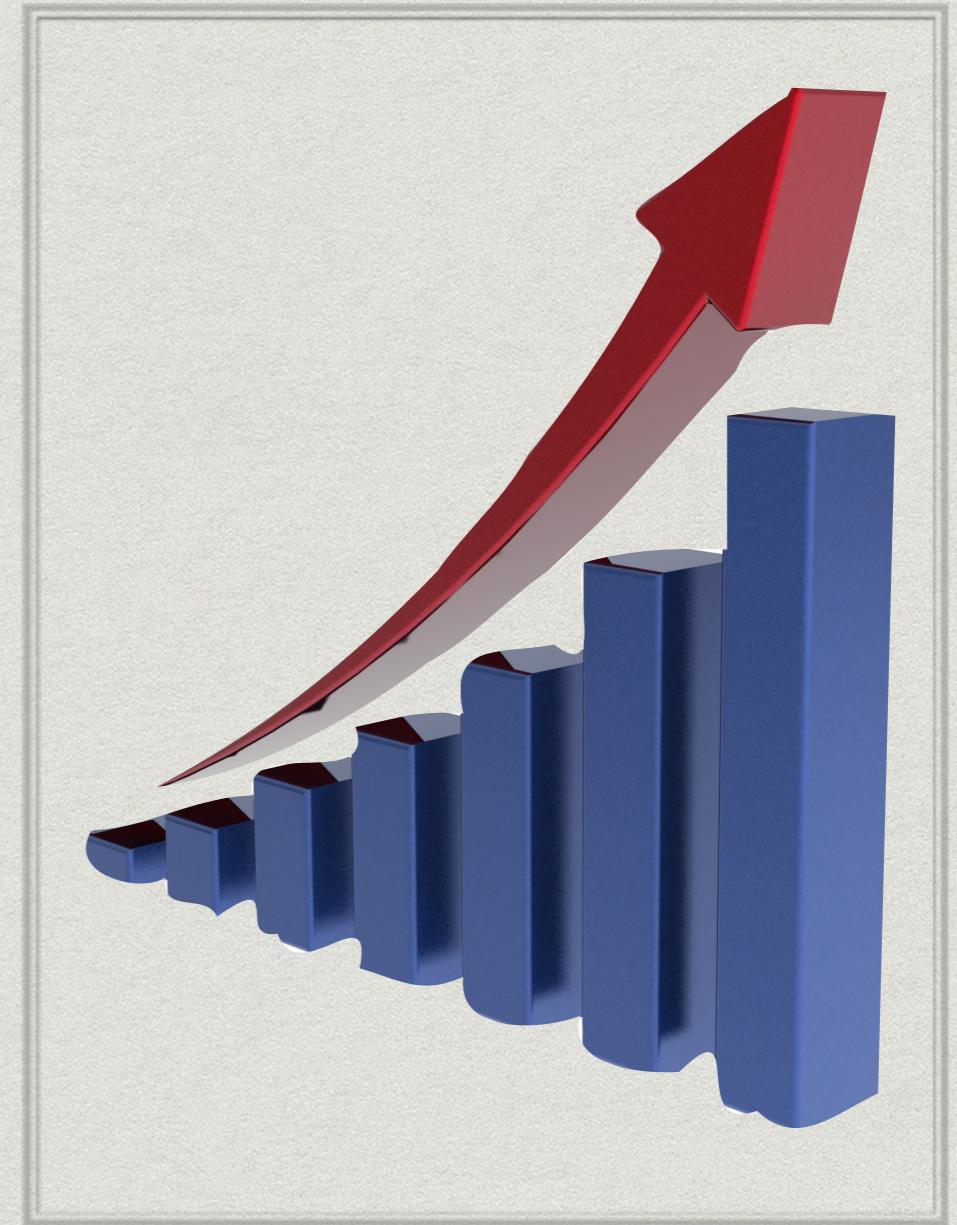
WordPress plugin'as

Sintetinis augimas: 20x

Realus augimas: 10x

Kodo pokytis: 101%

Klasių pokytis: 132%



PROFILI AVIMO PAGRINDAI

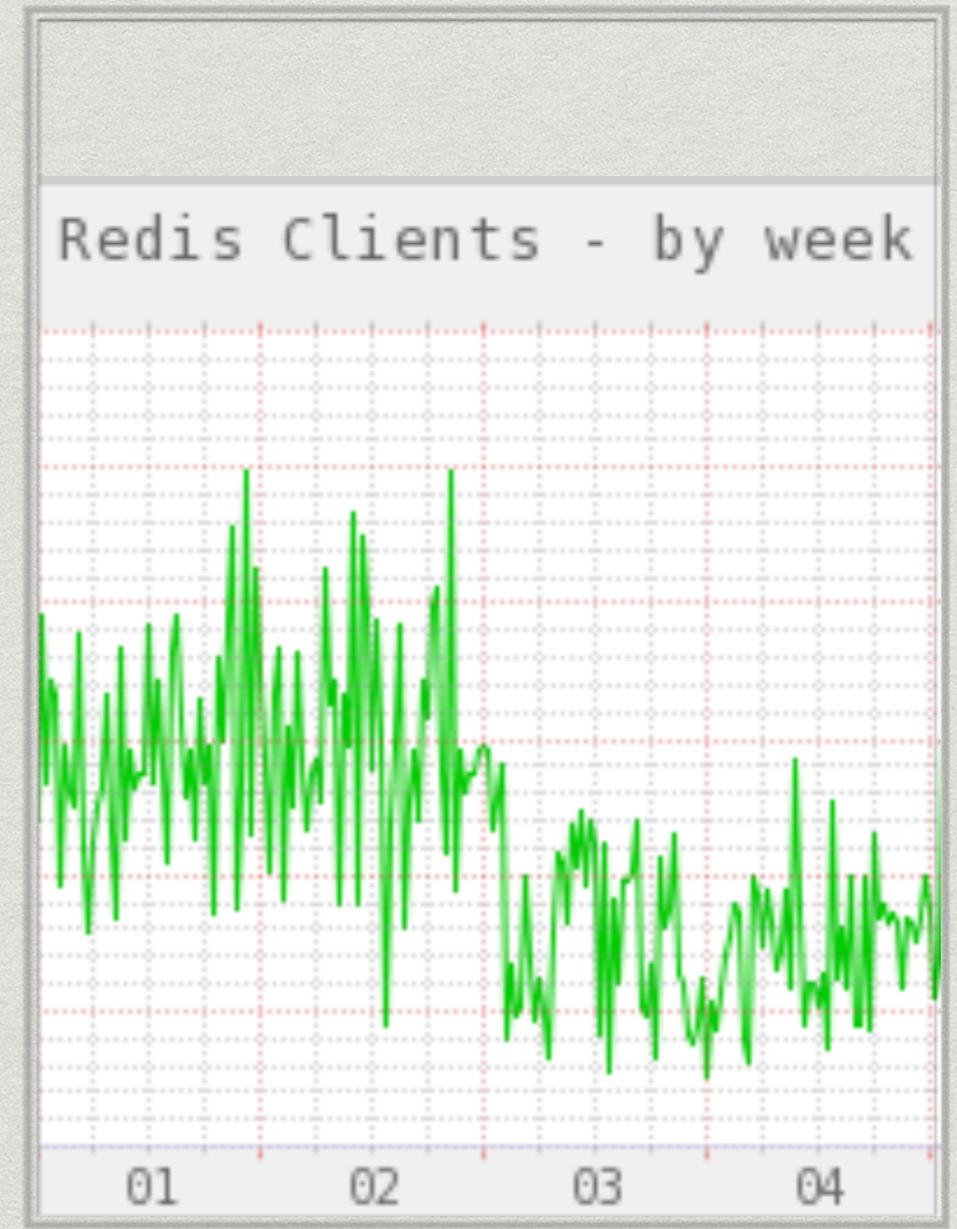


Apžvalga / stebėsena

Išteklių planavimas -- ką turime, ko reikės

Elgsenos pokyčių stebėjimas
-- rašymo ir skaitymo santykis,
cache panaudojimas

Atnaujintų sistemų problemas
-- klaidos, optimalumas



Priemonės: cacti, munin, nagios, zabbix

Auditorija: SysOps, sysadmin

Ižvalga / profiliavimas

Vykdymo kelio analizė

Išteklių naudojimo sekimas

Detalumas prieš greitą apžvalgą

Priemonės: xdebug, xhprof, Zend Debug (xhprof)

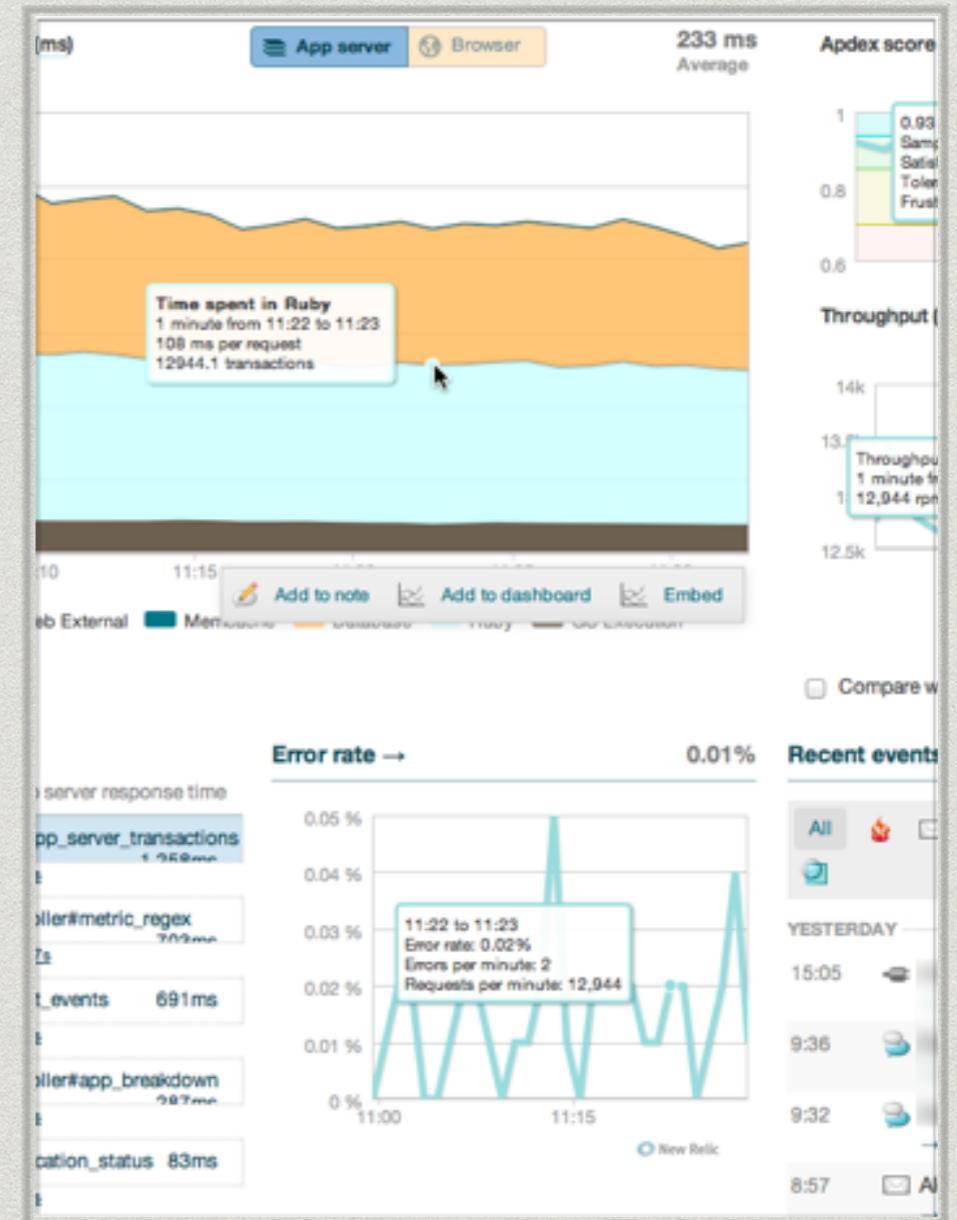
Auditorija: programuotojai, architektai

Integruoti sprendimai

Apžvalgos ir ižvalgos derinys

Apima daugiau nei vieną sistemą

Dalinis (konfigūruojamas) profiliavimas veikiančioje sistemoje



Priemonės: NewRelic, AppDynamics, Compuware

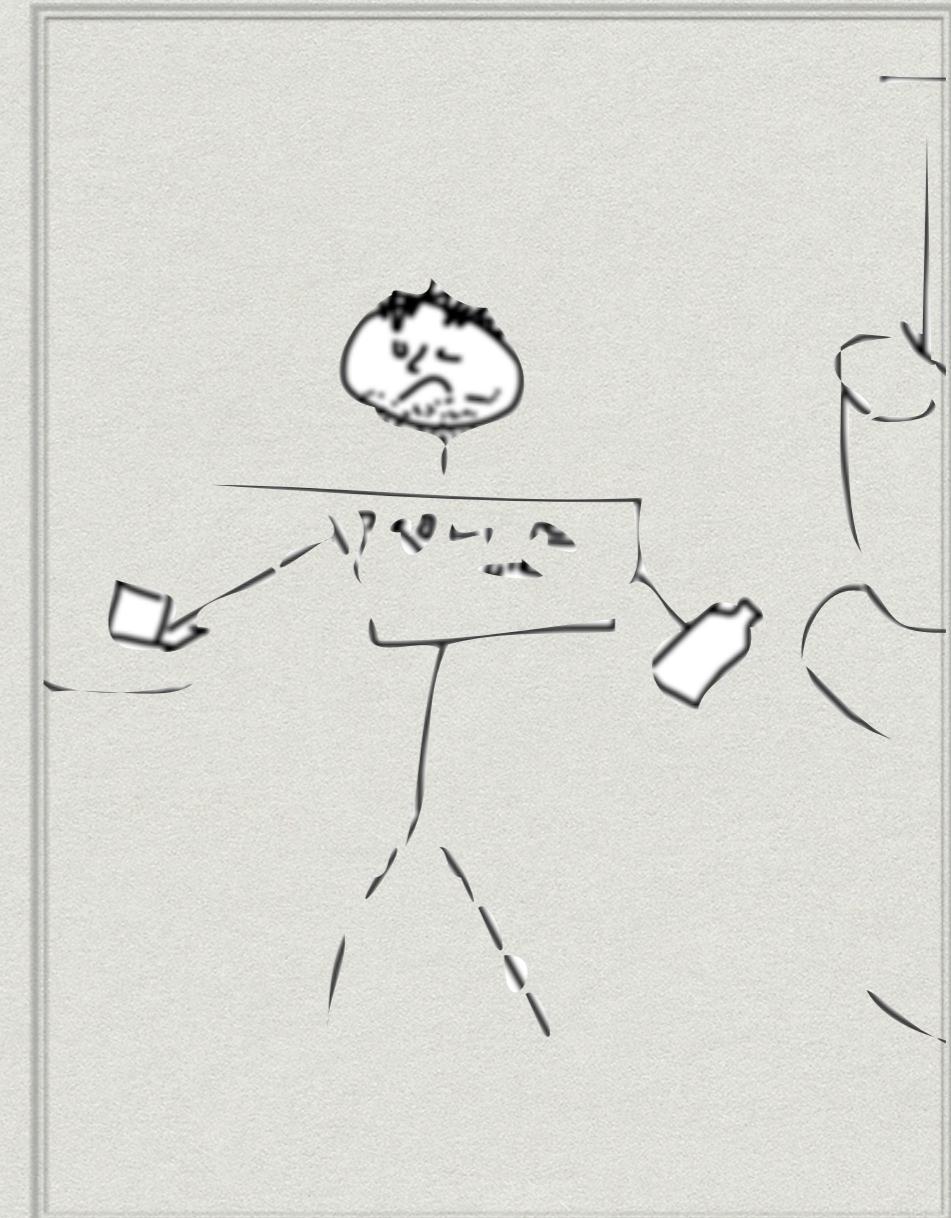
Auditorija: visi ankstesni

PMP

Ižvalga į konkrečią posistemę
realiu laiku

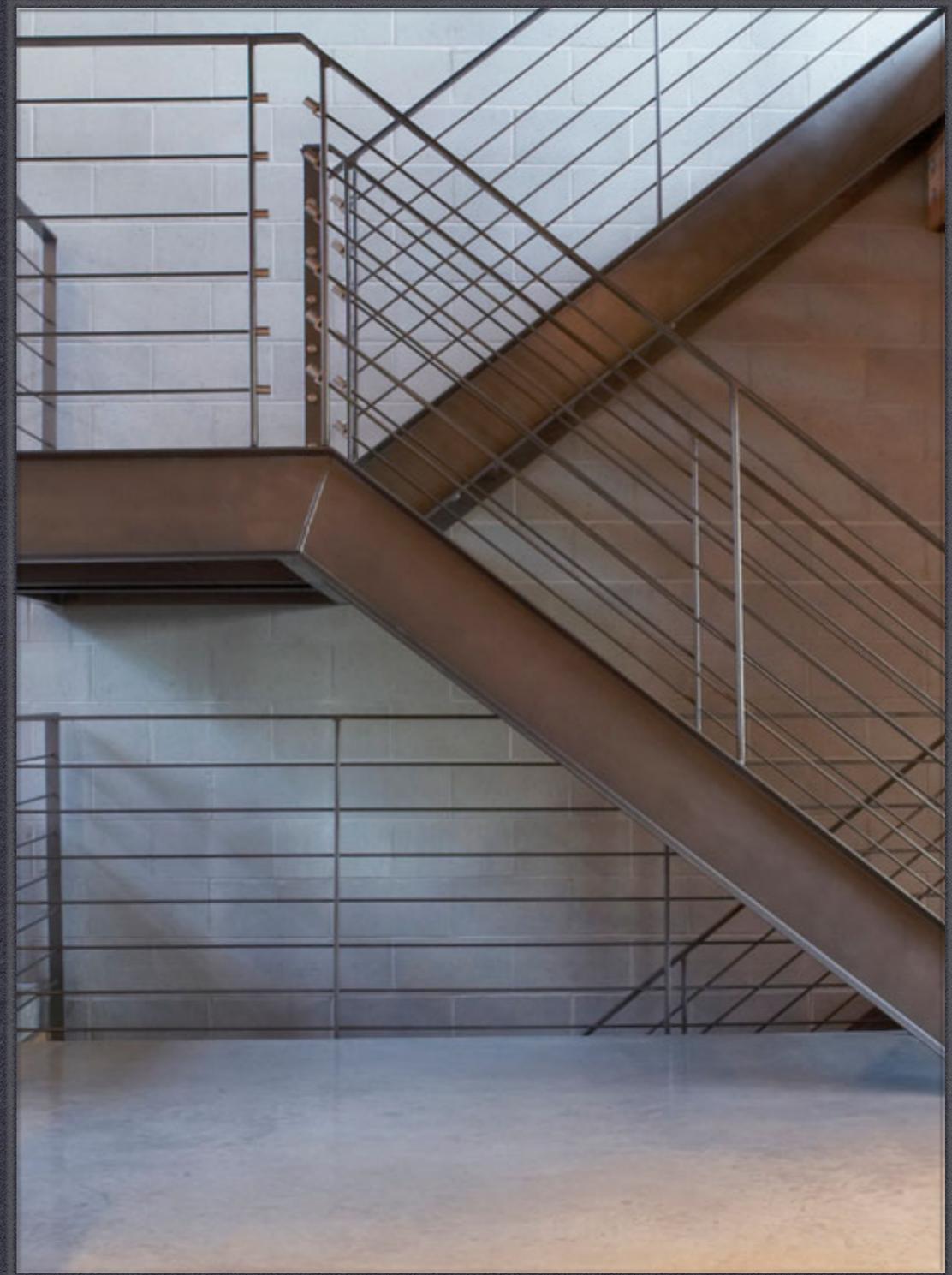
Būtinos sisteminės žinios

Dažniausiai: zero set-up time



ARCHITEKTŪRA

TIKSLAI IR PRINCIPAI



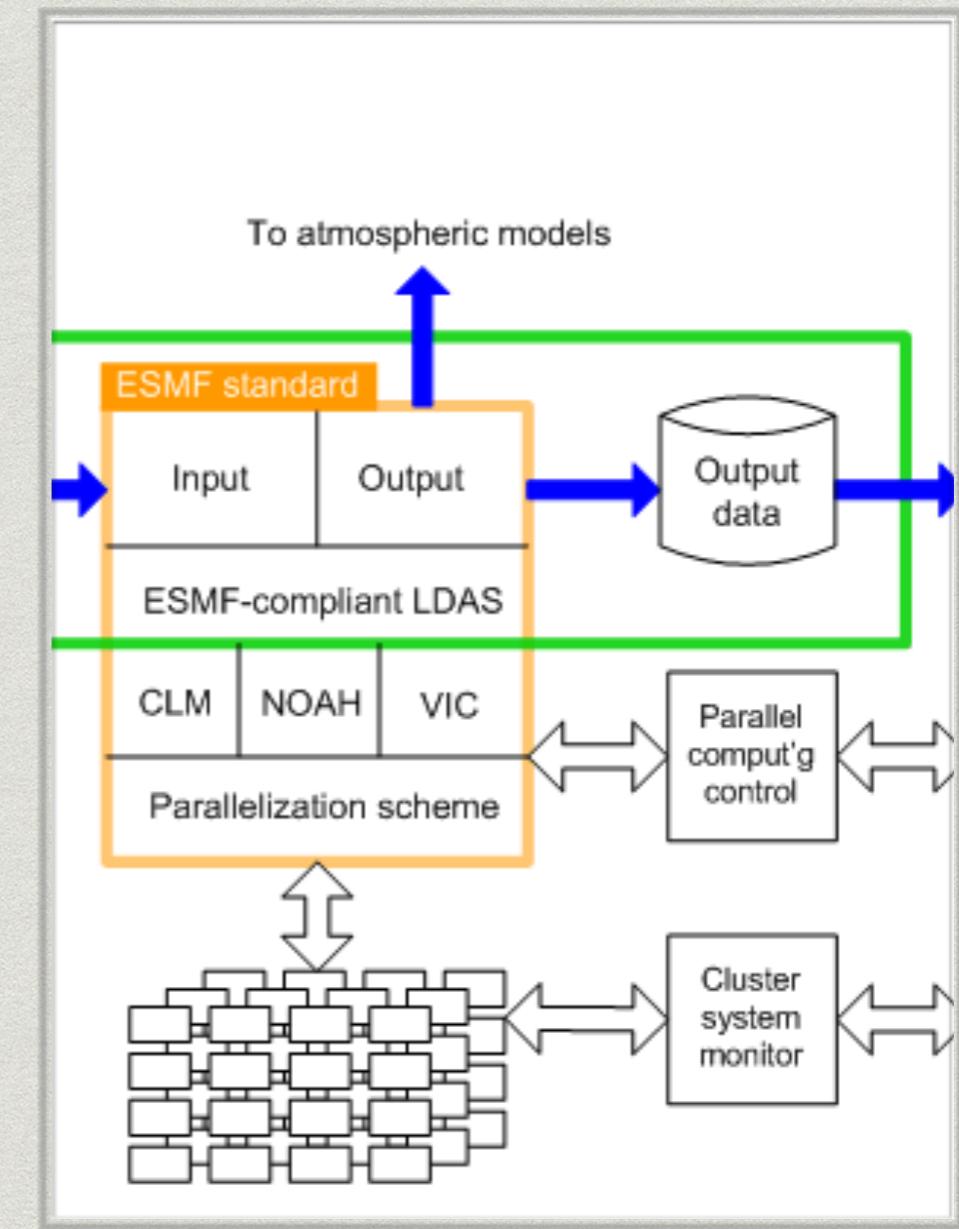
Informacijos surinkimas

Verslo ryšiai

Vidinių sistemų ryšiai

Numatomos aplinkos

Duomenų struktūra



Plano parengimas

Aptiki galimybių (gebėjimų) trūkumus

Funkcinio vystymosi kelio numatymas

Atsižvelgiant į apribojimus (atmintis, ciklai, kt.)

Numatyti istorinio suderinamumo palaikymą

Fizinis lygmuo

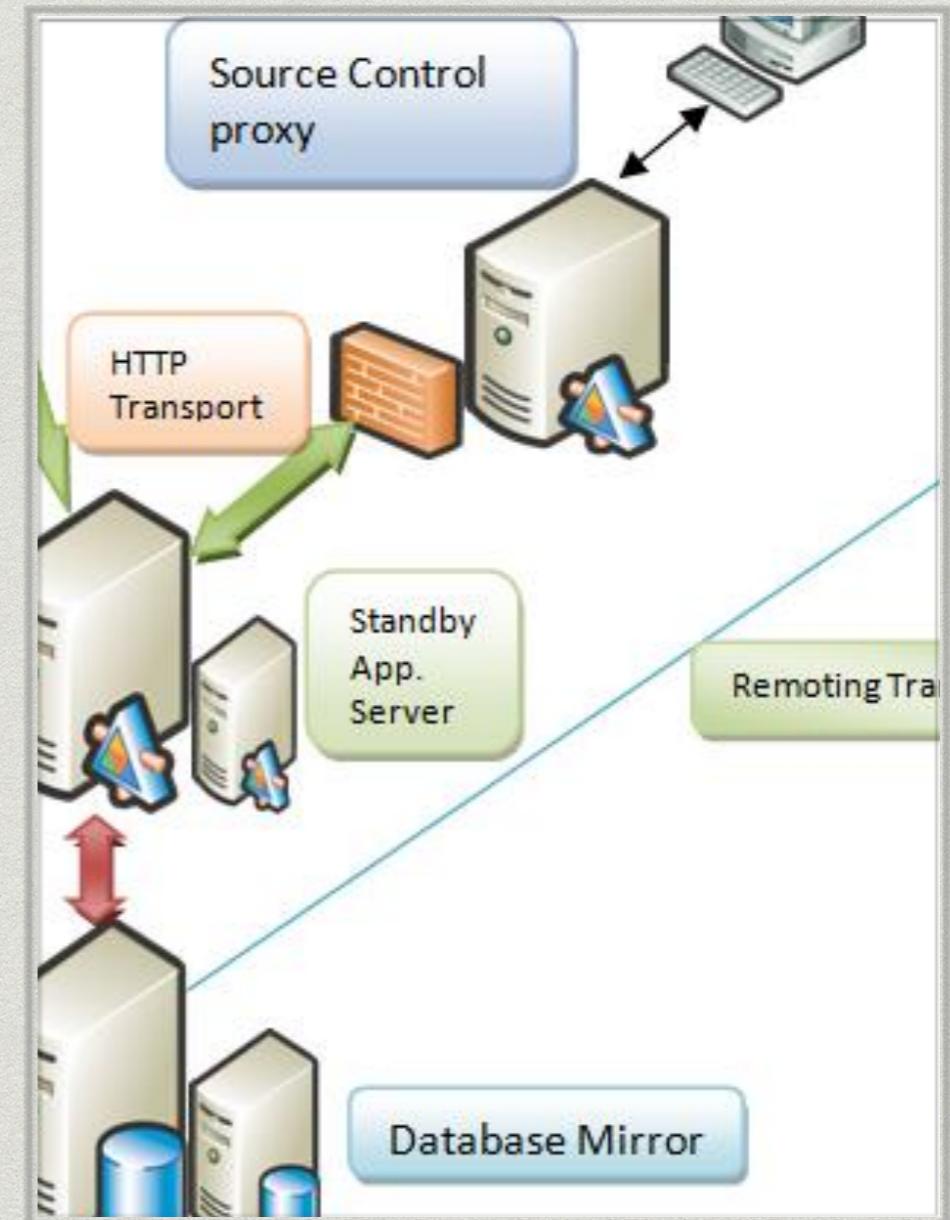
Prieigos būdai

Saugumo specifika

Atsparumo reikalavimai

Apkrovos prognozės

Sistemos specifika



Abstrakcijos

Istorinis suderinamumas -- savo sistemos,
programinės aplinkos, naudojimo sekų

Adaptyvus išteklių naudojimas -- galimos
bibliotekos, funkcionalumas pagal poreikius

OOP pradžiamokslis

NAŠUMO REIKŠMĖ ARCHITEKTŪRAI



Postulatai

Erdvė ir laikas yra visuomet funkciškai susiję

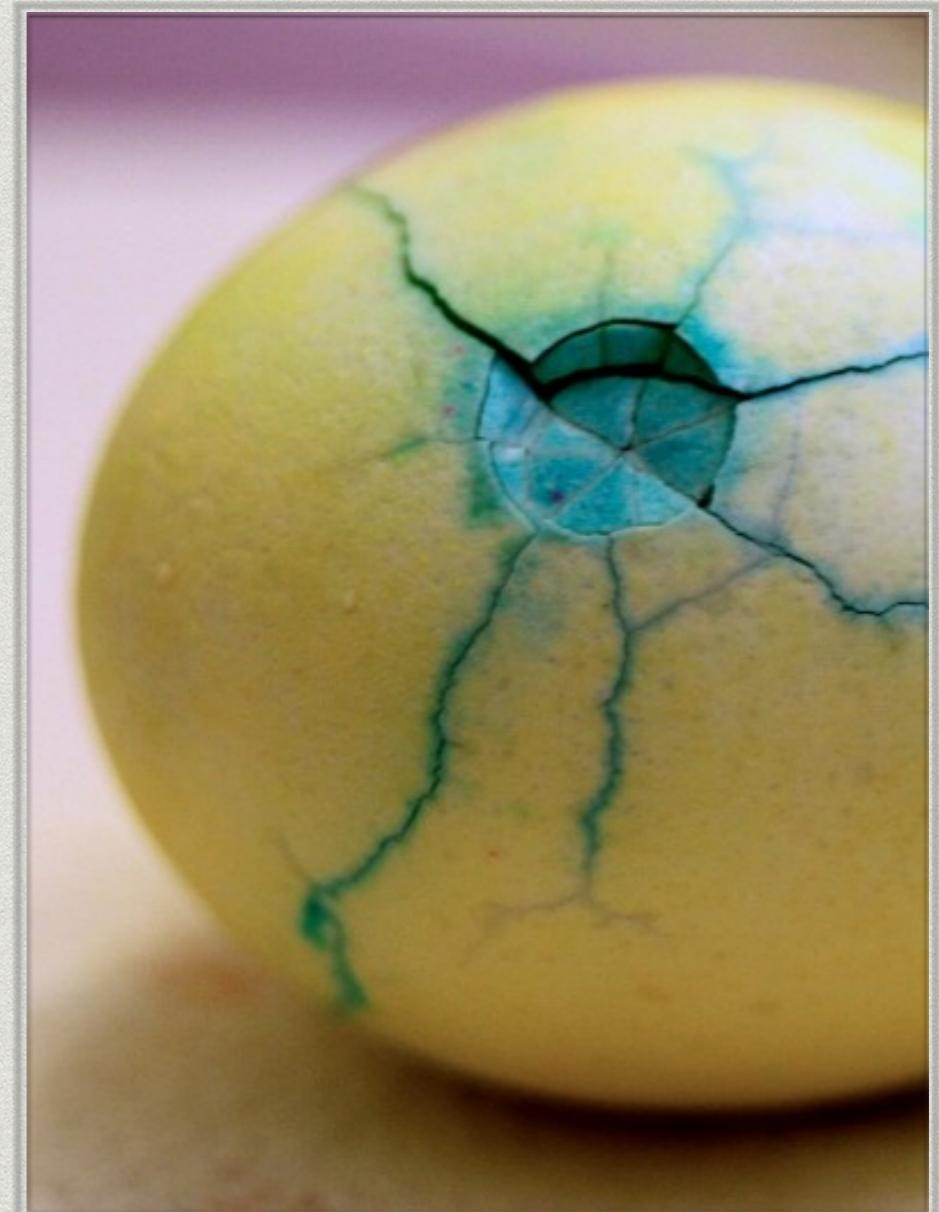
Net dedikuotoje architektūroje *cache* nėra panacėja

Visada daugiausia laiko užima I/O operacijos

Netikslingas profiliavimas lemia nevertingas mikrooptimizacijas

Nesisteminis požiūris

1. Sudėti viską į *cache*
 - 1.1. Sutvarkyti *invalidavimą*
 - 1.1.1. Netrinti visko iškart
 - 1.1.1.1. Padaryti, kad veiktų greitai
 - 1.1.1.1.1. Paskutinis *patch'as*



Sisteminis požiūris

Kūrimo pradžioje našumas apibrėžiamas

Tikrinama periodiškai ir dalimis

Paliekamas rezervas pradiniuose reikalavimuose

Dalinamasi našiais sprendimais

ATVEJO ANALIZĒ



Pūtimas dėl našumo

Vykdymo tarpininkas -- bendrinis, turi būseną

MVC pyktis -- klasė veiksmui, visi turi būti ploni

Išmanūs primityvai -- trumpi šuoliai, atmintis prieš laiką

Klasika: late binding, acquisition on demand,
registries over singletons, passing by reference

Bendri reiškiniai

Kartojamos operacijos

Nenaudojamų failų apdorojimas vs. autoload

Ilgų I/O rezultato nesaugojimas

Pernelyg specifiniai duomenys *cache*

Duomenų dalinio išrinkimo problema

Priminimai / mikro opt.

Domėtis standartine biblioteka (bonus points:
suprasti kaip veikia kontrolės elementai)

Prieštaros: \$GLOBALS, regex XML, value entities

Taupyti kompiliatoriaus žinias (neapibrėžti kvietimai,
argumentų analizė)

serialize | JSON | igbinary

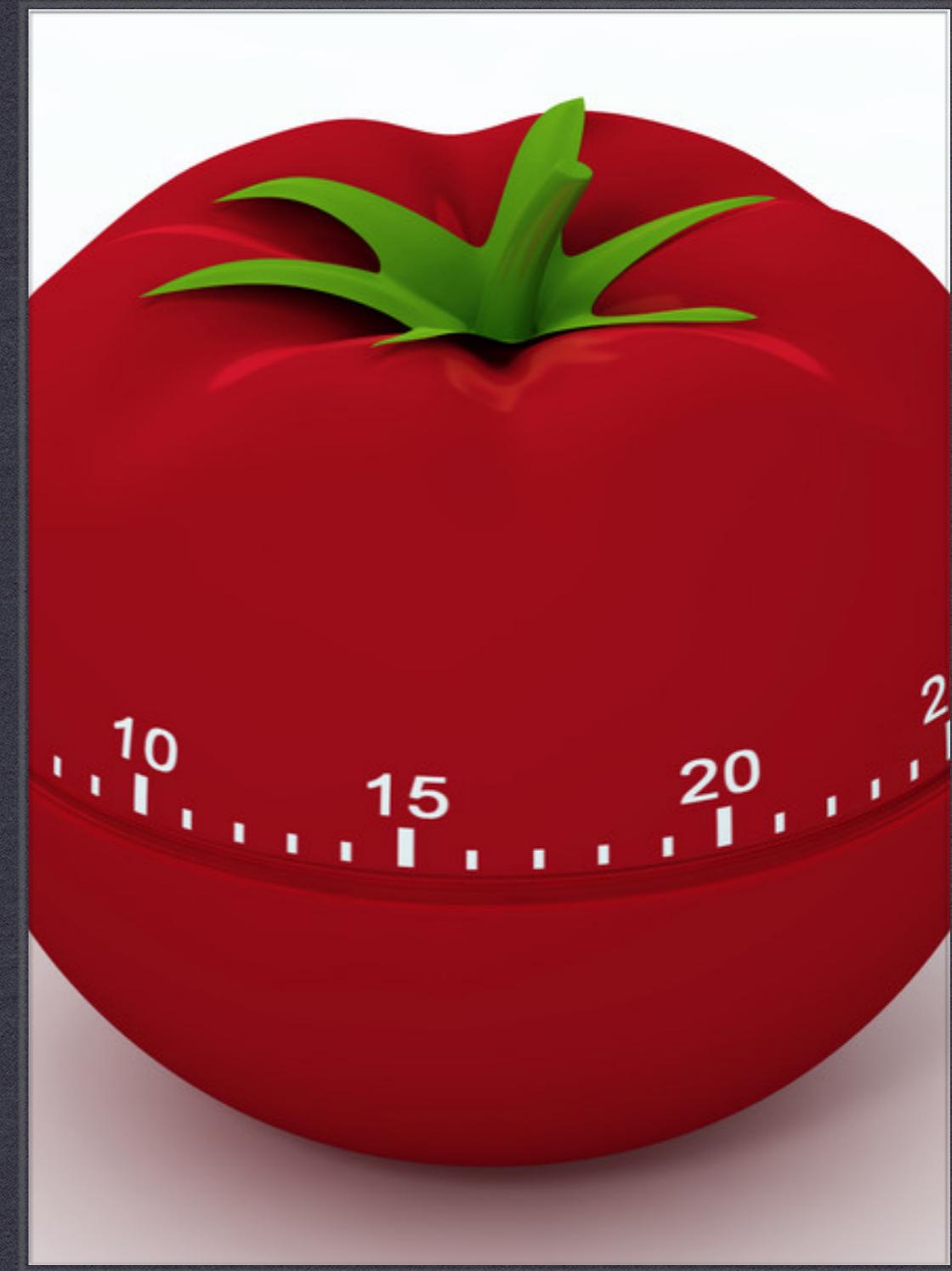
WordPress pyktis

Abstrakcija suderinamumui

Veiksmai ir filtrai

Reikšmių saugojimas (option, transient, meta)

REZULTATAS



Etapai

Savininkas nustato lūkesčius (kaip vartotojas)

Architektas numato išteklius ir priemones

Programuotojas naudoja geriausias priemones

Operatorius stebi panaudojimą ir registruoja pokyčius



Justas Butkus | jbutkus@time.ly