



СибГУТИ

СИБИРСКИЙ
ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ
И ИНФОРМАТИКИ

Инструменты Эксплуатации

План занятия

Теория

- Операционные системы
- Мониторинг
- Средства Диагностики
- Bash и другие скриптовые языки программирования

Практика

- 1) Установить СУБД (postgres или mysql(mariadb))
- 2) Установить веб сервер (apache или nginx, php-fpm,php)
- 3) Сделать Инвентаризацию в доме и погрузить данные в Субд + парсим результат в веб

Что это вам даст

- Понимание инструментов и средств Эксплуатации
- Узнать о средствах диагностики
- Как использовать скриптовые языки программирования

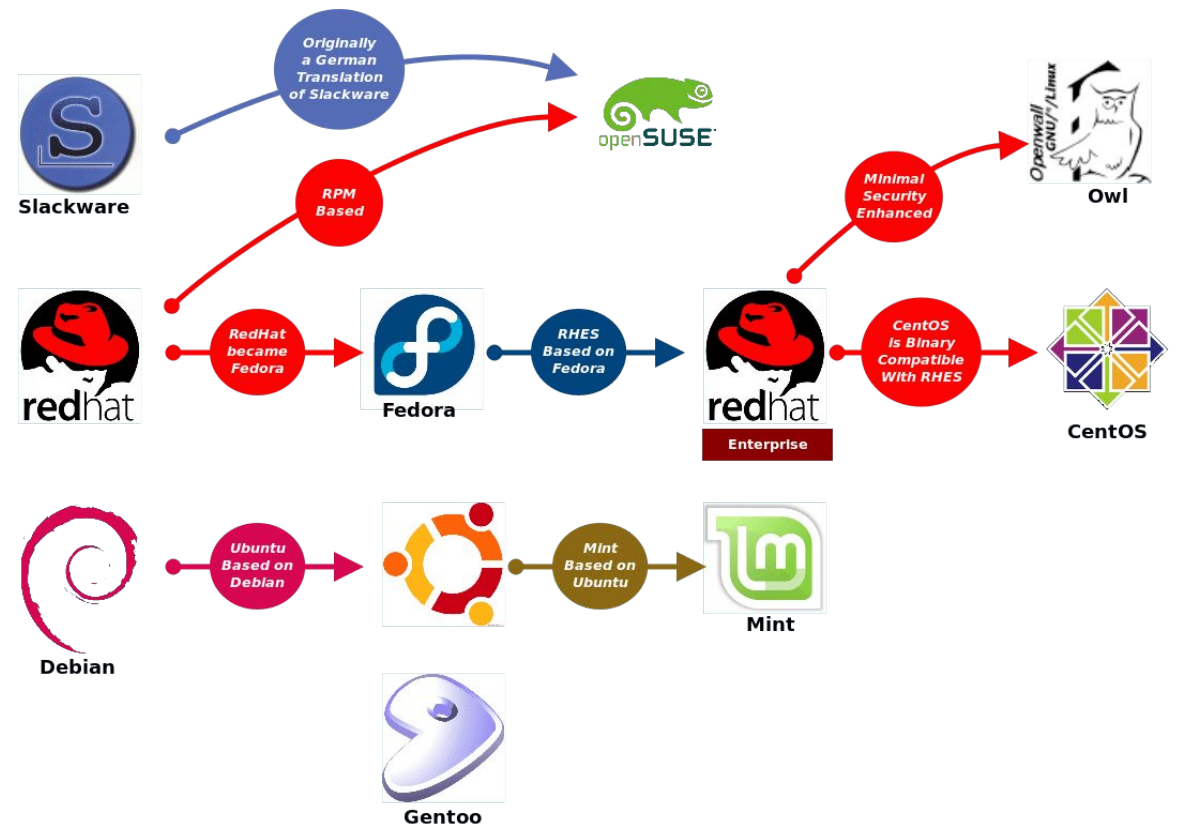
Операционные системы

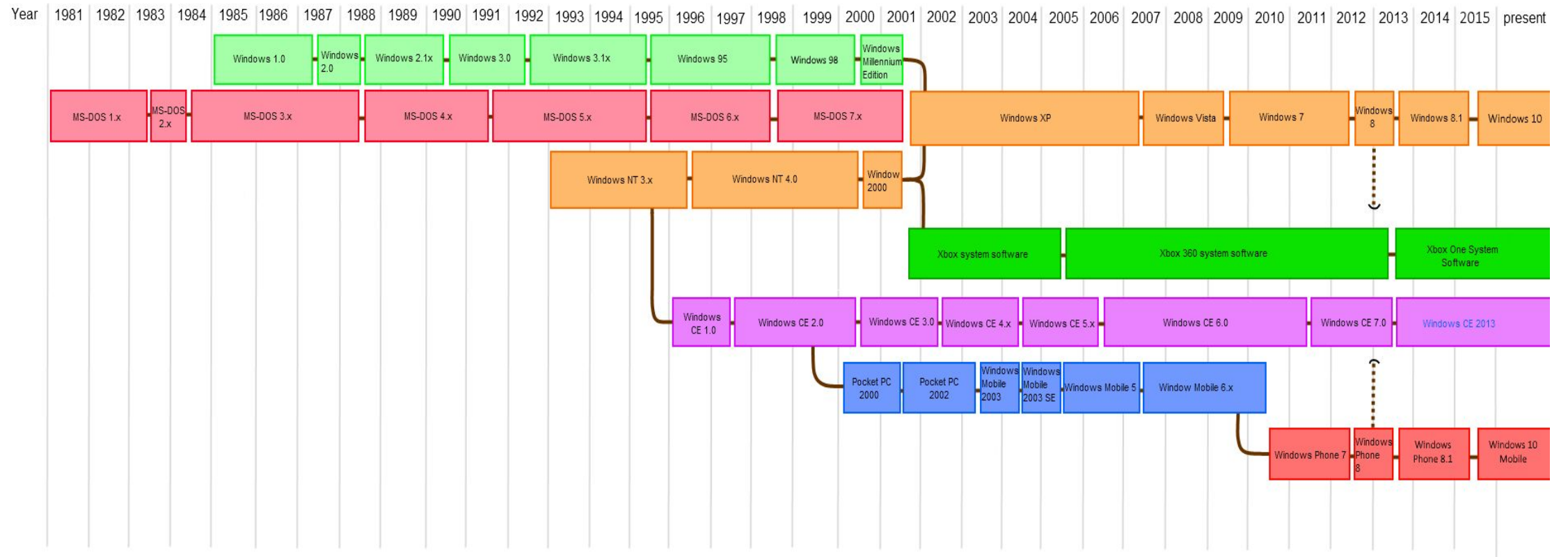




Дистрибутивы Linux

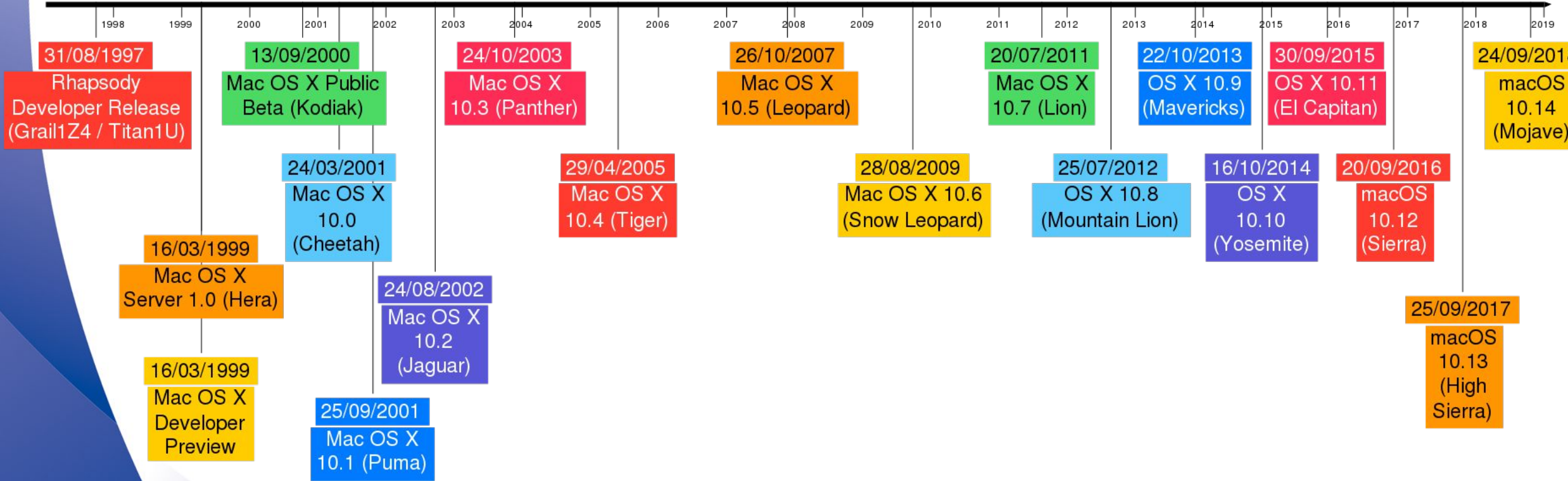
Как развивалось и множилось





1997

2019





Lollipop: 32%

Marshmallow: 31.20%

Kitkat: 18.80%

Jelly Bean: 9.10%

Nougat: 7.10%

Gingerbread: 1%

Ice Cream Sandwich: 0.80%

Android Versions' Relative Distribution



Сравнение производительности ОС

1) Создание и удаление директории

- Во время теста создётся большое количество вложенных директорий, а затем удаляется.

Операции проводятся встроенным в ОС средствами: mkdir/rmdir/rm

	Linux	Win	Win+AVP	MacOS
Создание (сек.)	37	137	137	46
Удаление (сек.)	2	447	451	3



2) Синтетический тест, с использованием MYSQL

Тесты проводились по 3 раза без заметной разницы, количество созданных в mysql записей равно 237201, на компьютерах

Debian Intel(R) Core(TM)2 Duo CPU E7300 @ 2.66GHz

Ubuntu Intel(R) Celeron(R) CPU E1200 @ 1.60GHz

WinXP Home Intel(R) Celeron(R) CPU E1200 @ 1.60GHz

Для проведения теста необходимы mysql сервер и клиент, интерпретатор python

Суть теста:

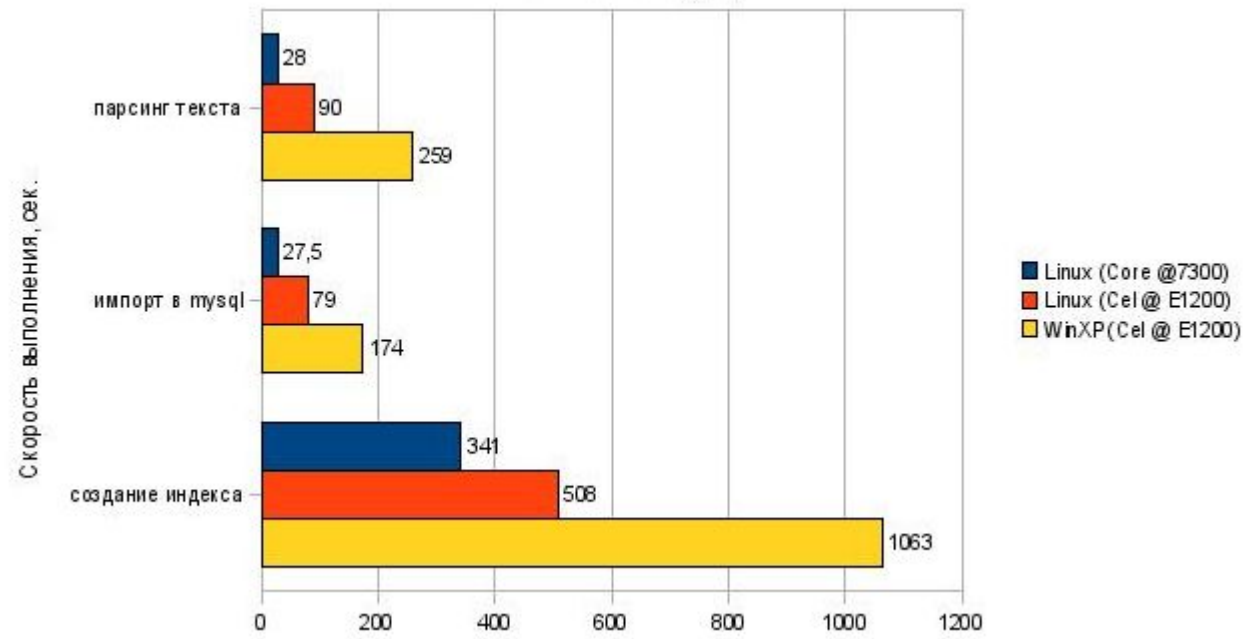
распарсить текстовый файл с данными о файловой системе на внешнем ftp сервере, создать 3 таблицы: файлов, путей и расширений, сгенерировать sql файл для таблицы файлов

вставить все данные в mysql

создать индекс

	Linux (Duo CPU E7300)	Linux (Cel @ E1200)	Windows (Cel @ E1200)
парсинг txt (сек)	28	90	259
import mysql (сек)	27,5	79	174,5
создание индекса (сек)	341	509	1063

Работа с mysql



Мониторинг

ZABBIX

Nagios®



New Relic.



MUNIN

RRDtool

logging & graphing

pingdom

TOOLS



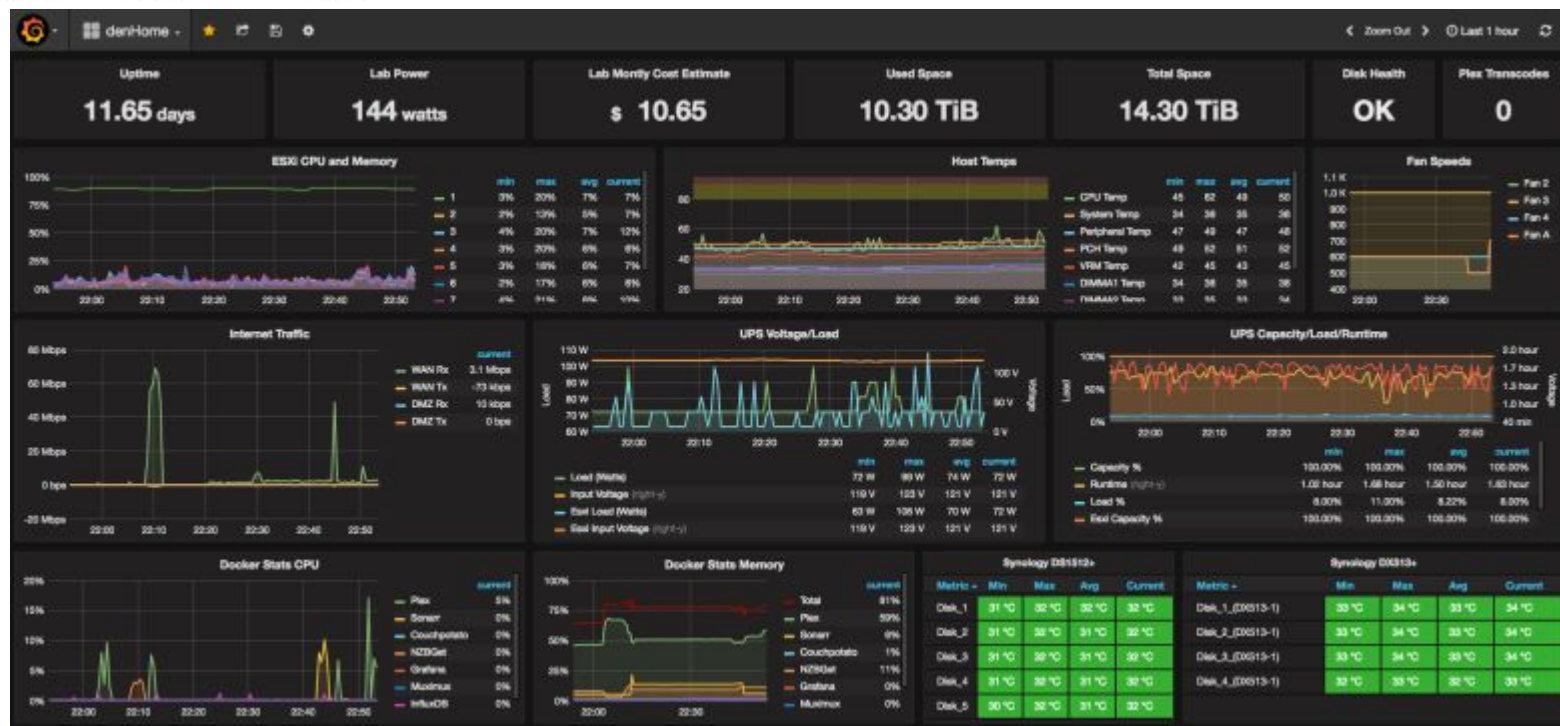
elasticsearch



logstash



kibana



ЗАЧЕМ ВСЁ ЭТО НУЖНО???

Раньше:

Мониторились, в основном, системные показатели: CPU, память, диски, сеть. Этого вполне хватало, потому что там крутилось одно приложение на php, и ничего больше не использовалось. Проблема в том, что по таким показателям обычно мало что можно сказать. Либо работает, либо нет. Что именно происходит с самим приложением, выше уровня системных показателей понять сложно.

Если проблема была на уровне приложения (не просто “сайт не работает”, а “сайт работает, но что-то не так”), то клиент сам писал или звонил, сообщал, что есть такая-то проблема, мы шли и разбирались, потому что сами мы такие проблемы заметить не могли.

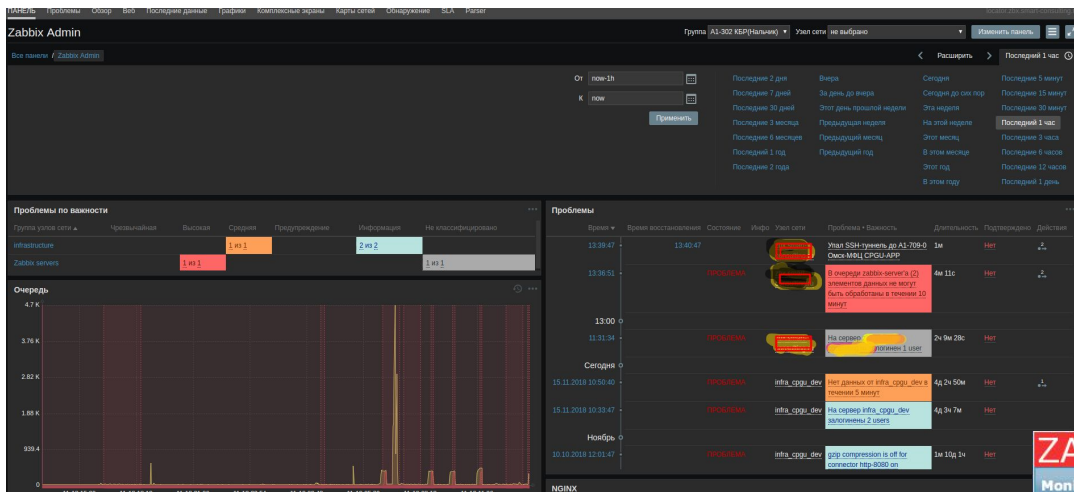
Сейчас:

Надо мониторить не только дискретное “работает/не работает”, а гораздо больше градаций. Что, в свою очередь, позволяет ловить проблему до того, как все рухнет.

Кроме того, теперь надо следить и за бизнес-показателями.

Почему?

Усложнение самих систем, конечно, повлекло за собой большее количество возможных проблем. Появились метрики приложений, количество запущенных тредов у Java application, частота garbage collector pauses, количество событий в очереди. Очень важно, чтобы мониторинг также следил за масштабированием систем. Допустим, у вас Kubernetes HPA. Надо понимать, сколько запущено подов, и с каждого запущенного пода должны идти метрики в систему мониторинга приложения, в арт.



ZABBIX

[Help](#) | [Get support](#) | [Print](#) | [Profile](#) | [Logout](#)

Monitoring | Inventory | Reports | Configuration | Administration

[Dashboard](#) | [Overview](#) | [Web](#) | [Latest data](#) | [Triggers](#) | [Events](#) | [Graphs](#) | [Screens](#) | [Maps](#) | [Discovery](#) | [IT services](#)

History: [トリガーの設定](#) » [ダッシュボード](#) » [ユーザープロファイル](#) » [Dashboard](#) » [Overview](#)

PERSONAL DASHBOARD

Favorite graphs

- [vSphere 001: CPU utilization](#)
- [vSphere 002: CPU utilization](#)
- [vSphere 003: CPU utilization](#)

Favorite screens

- [Zabbix server performance](#)
- [JBoss performance](#)
- [Oracle RAC](#)
- [Network map](#)

Favorite maps

- [Network devices](#)
- [VMWare production](#)

Status of Zabbix

Parameter	Value	Details
Zabbix server is running	Yes	localhost:10051
Number of hosts (monitored/not monitored/templates)	85	47 / 0 / 38
Number of items (monitored/disabled/not supported)	502	493 / 0 / 9
Number of triggers (enabled/disabled) [problem/ok]	291	291 / 0 [10 / 281]
Number of users (online)	2	1
Required server performance, new values per second	7.7	-

Updated: 02:41:40 AM

System status

Host group	Disaster	High	Average	Warning	Information	Not classified
Business System	0	0	0	0	0	0
Clouds	0	0	0	0	0	0
Database servers	0	0	0	0	0	0
JBoss instances	0	0	0	3	0	0
Network Devices	0	0	0	0	0	0
Private Cloud	0	0	0	5	0	0
Web servers	0	0	0	0	0	0
Zabbix servers	0	0	0	2	0	0

Updated: 02:41:41 AM

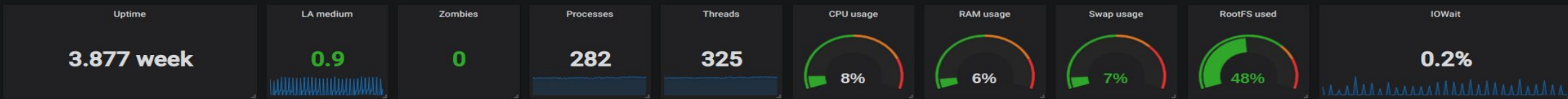
Host status

Host group	Without problems	With problems	Total
Business System	17	0	17
Clouds	2	0	2
Database servers	2	0	2
JBoss instances	0	3	3

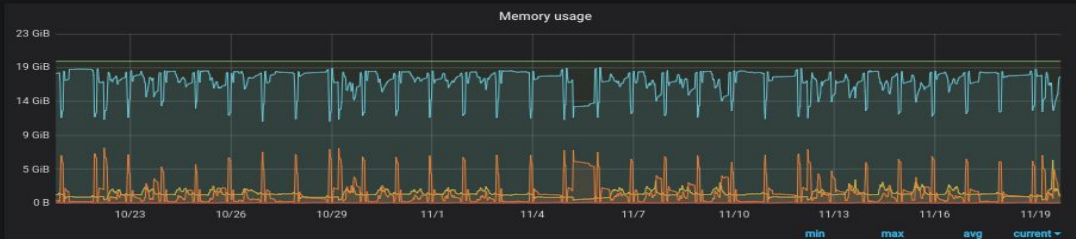
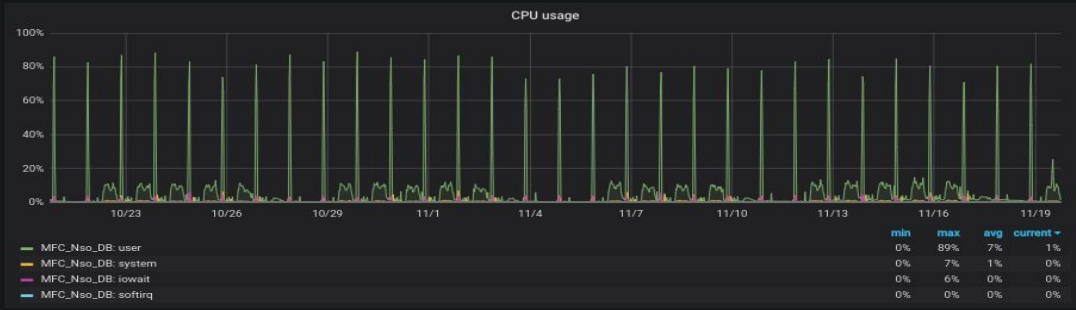


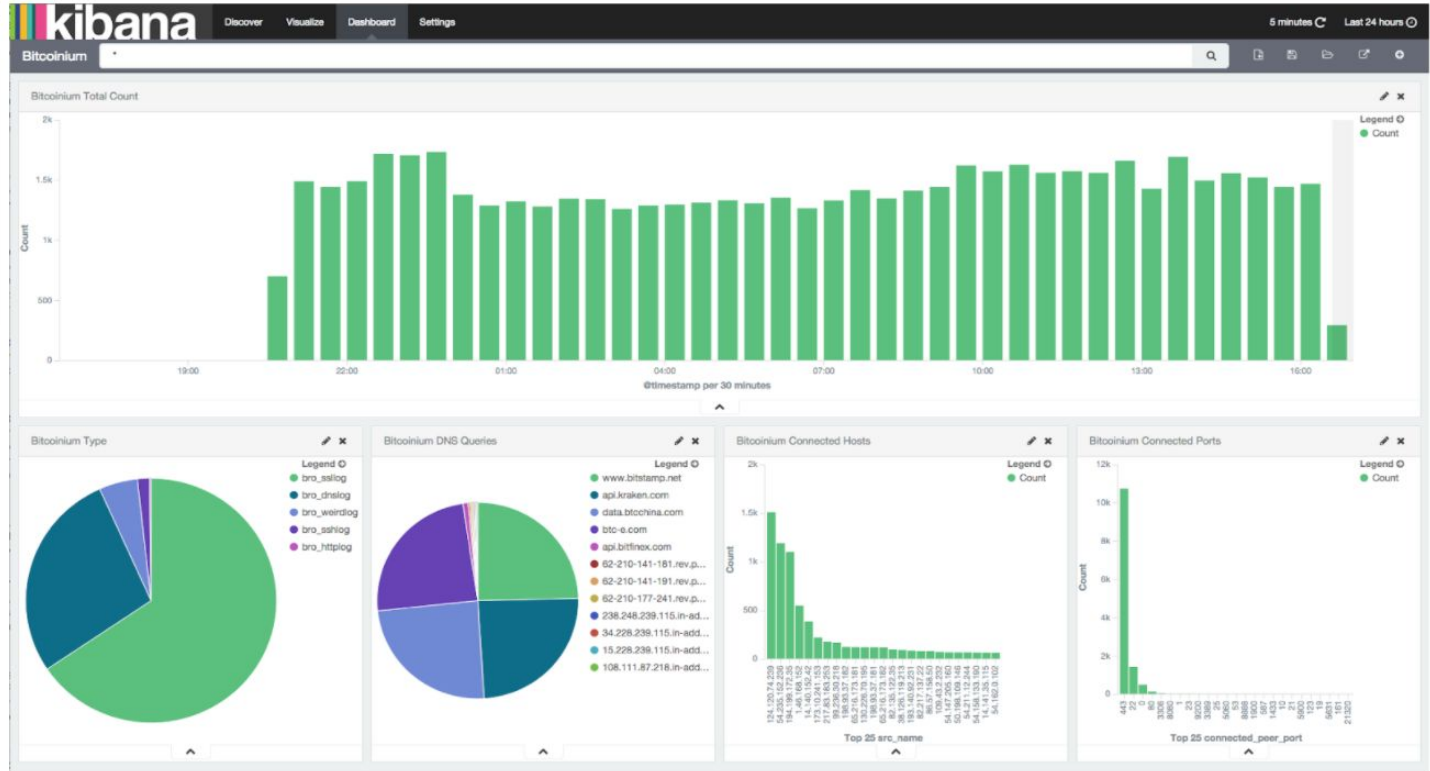
21.0 GB 12 2.147 GB 16.9 GB 211 GB

Quick overview



CPU





Средства Диагностики

Материалы:

<https://habr.com/company/ua-hosting/blog/281519/>

#!/Bash

<https://linuxconfig.org/bash-scripting-tutorial>

Пример

```
#!/bin/bash
```

```
echo -e "Hi, please type the word: \c "
```

```
read word
```

```
echo "The word you entered is: $word"
```

```
echo -e "Can you please enter two words? "
```

```
read word1 word2
```

```
echo "Here is your input: \"$word1\" \"$word2\""
```

```
echo -e "How do you feel about bash scripting? "
```

```
# read command now stores a reply into the default build-in variable $REPLY
```

```
read
```

```
echo "You said $REPLY, I'm glad to hear that! "
```

```
echo -e "What are your favorite colours ? "
```

```
# -a makes read command to read into an array
```

```
read -a colours
```

```
echo "My favorite colours are also ${colours[0]}, ${colours[1]} and ${colours[2]}:-)"
```

#!/usr/bin/python

```
#!/usr/bin/python3
```

```
import os
import subprocess
import paramiko
import sys
import re
import time
```

```
#Подключение по ssh
```

```
host = "point"
```

```
port = 22
```

```
user='root'
```

```
secret= 'neskazhupassword'
```

```
#Команда для выполнения
```

```
monitoring = "curl localhost:8080/rest/monitoring/ -s | sed -e 's/<br>//g' | sed -e 's/<\b>//g' | sed -e 's/<br>/#/g' | sed -e 's/<b>//g' | sed -e 's/<html><body>//g' | sed -e 's/<\body><\html>//g' | sed 's/ /_/'
```

```
#Подключение
```

```
client = paramiko.SSHClient()
```

```
client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
```

```
client.connect(hostname=host, username=user, password=secret, port=port)
```

```
#Выполнение команды на удалённом сервере
```

```
stdin, stdout, stderr= client.exec_command(monitoring)
```

```
#Вывод команды
```

```
data = stdout.read() + stderr.read()
```

```
# Преобразование русского текста
```

```
data2 = data.decode('utf8')
```

```
#print (data2)
```

```
# Закрытие соединения
```

```
data3=re.findall(r'^# ]+', data2)
```

```
client.close()
```

```
#Список из проверок встречающихся в выводе мониторинга
```

```
check_code = ['FAIL','WARN']
```

```
#Проверка совпадений в строке
```

```
result = [s for s in data3 if any(xs in s for xs in check_code ) ]
```

```
print (result)
```