Сентябрь 2023

MONOPOLY. ONLINE

Когда 100% CPU ничего не значат

Не доверяйте доступному CPU

Станислав Флусов

Структура доклада

- Проблематика и цели данного доклада
- Физический уровень
  - Core vs Threads (технология Hyper Threading)
  - Режимы работы процессора CPUFreq Governors
  - Гибридная технология Big.LITTLE
- Виртуализация
  - Технология VMWare vMotion
  - Правила DRS
  - Переподписка вычислительных ресурсов
  - Метрики производительности %WAIT, %RDY, %CSTP
- Контейнеризация
  - Ресурсы доступные внутри контейнера
  - Параметры reservation/limit для CPU и Memory
  - Вертикальное и горизонтальное масштабирование приложений

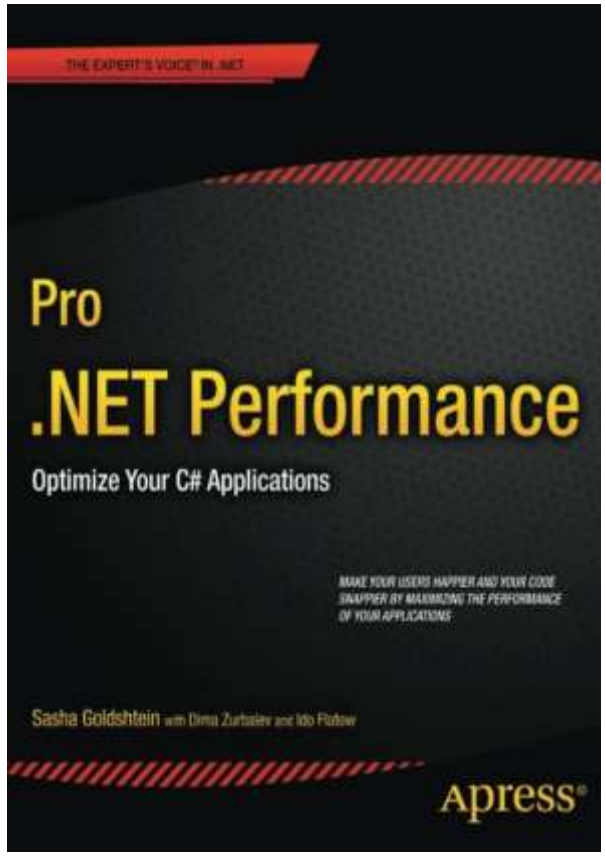
MONOPOLY.
ONLINE

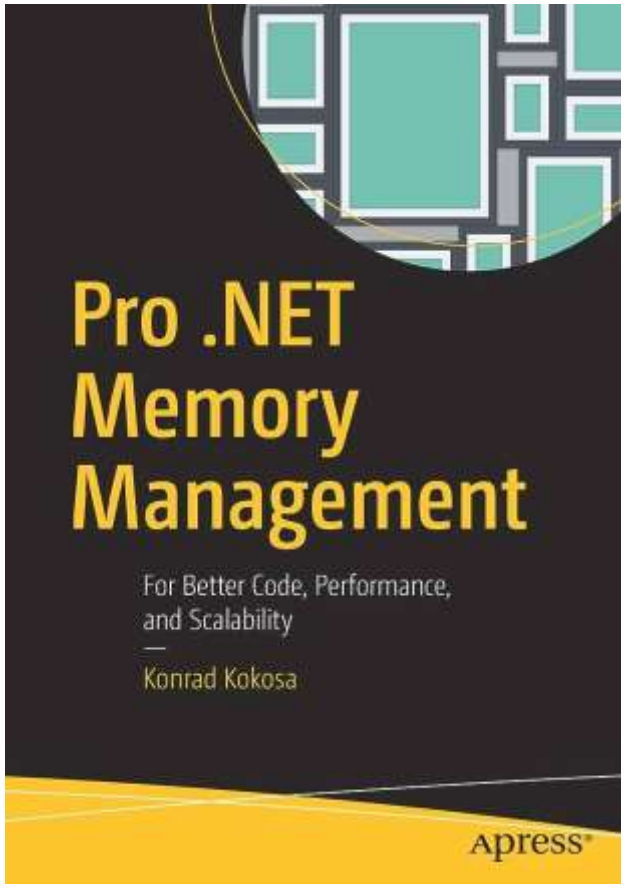
Проблематика и цели данного доклада

Проблематика

- Низкая скорость и стабильность деплоев на тестовых средах
- Низкая утилизация по CPU ~ 15 – 20%
- Периодическое зависание pods с их дальнейшим рестартом и частичной инвалидацией тестового сценария (crashloopback)
- Многократное повышение response time в сценариях, где такие значения кажутся странными

Вы пишите код в соответствии с лучшими практиками





https://www.amazon.com/Pro-NET-Performance-Optimize-Applications/dp/1430244585

https://www.amazon.com/Pro-NET-Memory-Management-Performance/dp/148424026X

Цели доклада

- Показать, что производительность находится не только на уровне кода приложения, существенная часть перфоманса находится именно на уровне инфраструктуры: виртуализации и контейнеризации

- Расширить знания бэкенд разработчиков о инфраструктуре, поскольку основную нагрузку на сервера генерируем именно мы (бэкенд разработчики)

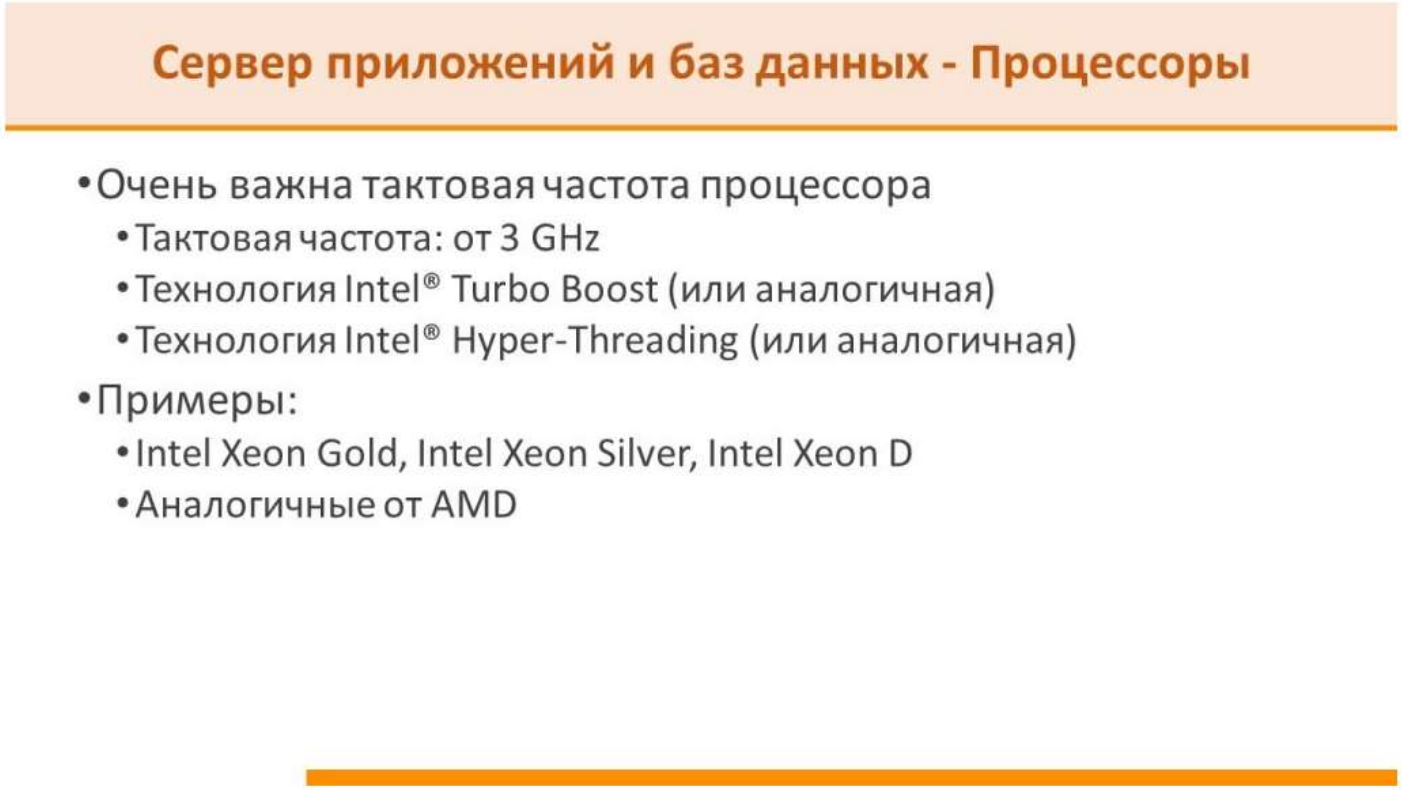
- Показать способы решения перфоманс ботленеков, связанных с инфраструктурой

MONOPOLY.
ONLINE

Физический уровень

Цитаты, которые невозможно забыть

- На моем локальном компьютере 1С ERP УХ работает намного быстрее, чем на выделенном сервере
- Необходимая производительность процессора от 3GHz, процессор 2.6 GHz недостаточно быстрый для 1С



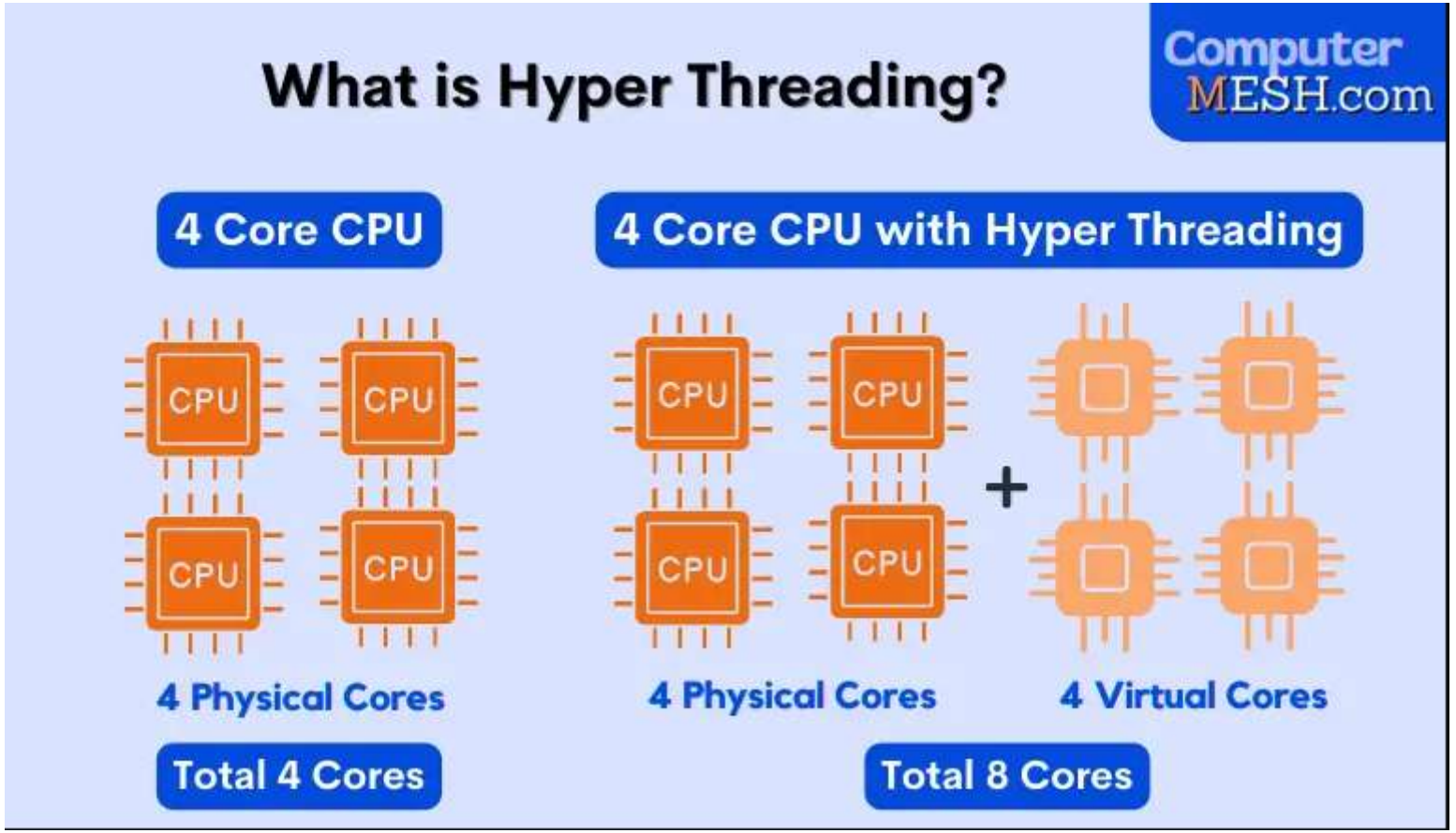
https://itbconsult.ru/blog/podbor-servera-dlya-1s-erp/

Неравный бой ноутбука и выделенного сервера

AMD Ryzen 7 5800H	
Description: with Radeon Graphics	
Class: Laptop	**Socket:** FP6
Clockspeed: 3.2 GHz	**Turbo Speed:** 4.4 GHz
Cores: 8 **Threads:** 16	**Typical TDP:** 45 W
TDP Down: 35 W	
Cache Size: L1: 512 KB, L2: 4.0 MB, L3: 16 MB	

Intel Xeon E5-2690 v4 @ 2.60GHz	
Description:	
Class: Server	**Socket:** FCLGA2011-3
Clockspeed: 2.6 GHz	**Turbo Speed:** 3.5 GHz
Cores: 14 **Threads:** 28	**Typical TDP:** 135 W
Cache Size: L1: 896 KB, L2: 3.5 MB, L3: 35 MB	
Memory Support: Max. Memory Size: 5.0 TB (DDR4 1600/1866/2133/2400, ECC Supported)	

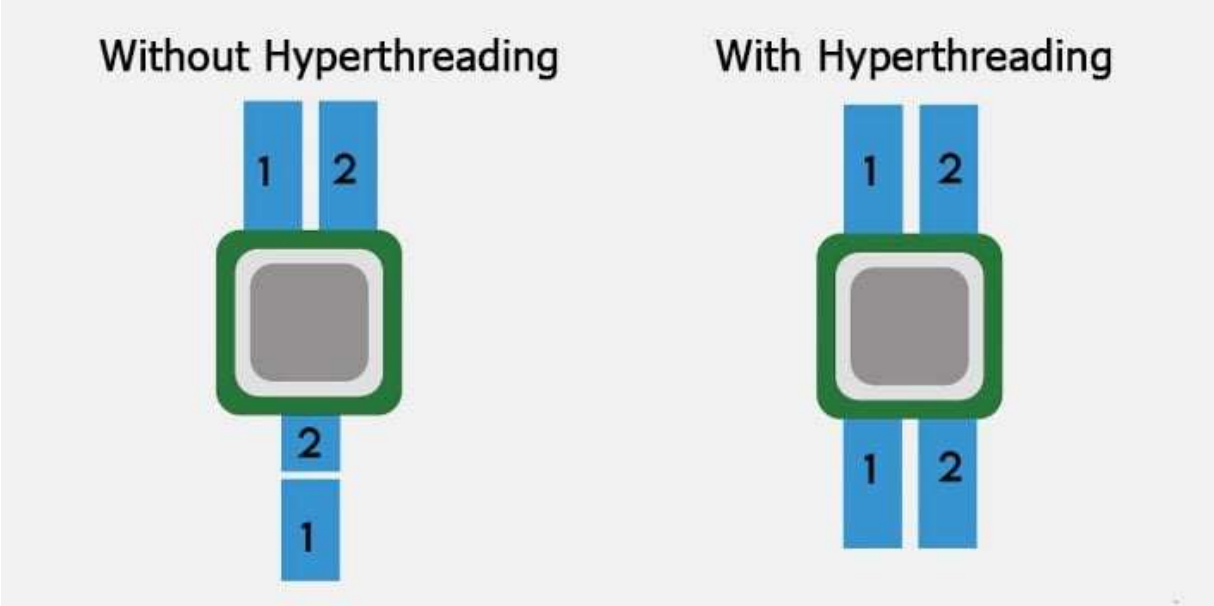
https://www.cpubenchmark.net/cpu.php?cpu=AMD+Ryzen+7+5800H



https://computermesh.com/what-is-hyper-threading-how-does-it-work/

Переподписка на уровне процессора – технология Hyper Threading

SMT - simultaneous multithreading



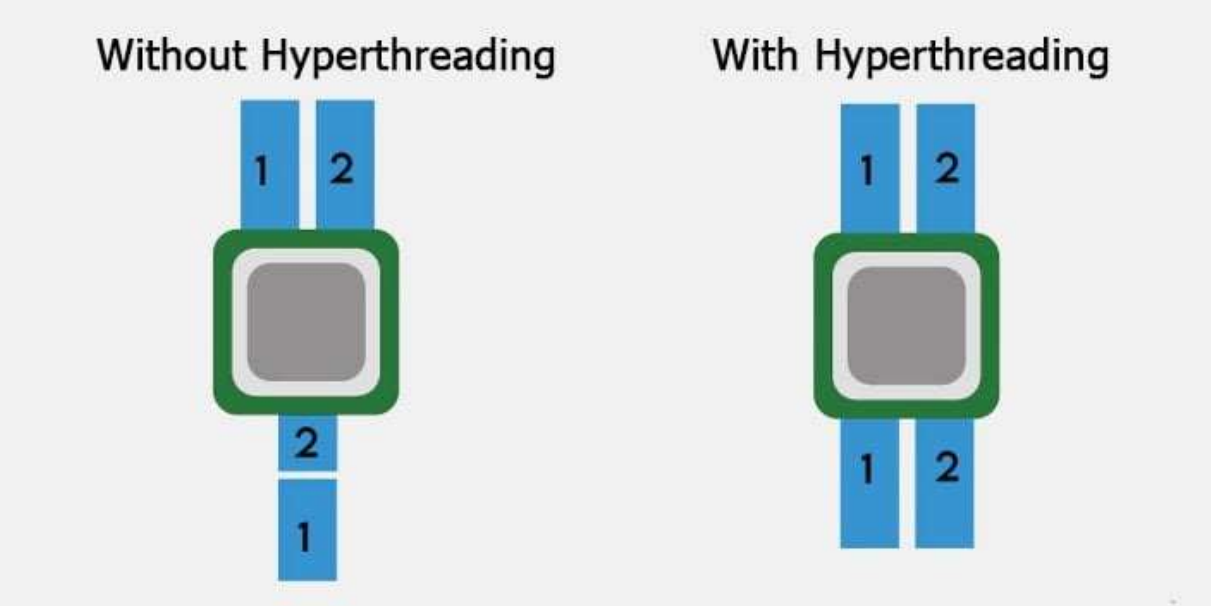
Принципы работы:

- Процессорное ядро одновременно хранит состояние двух потоков
- При наступления событий, процессор не простаивает в ожидании, а передает управление второму потоку
  - Промах при обращению к кэшу процессора
  - Выполнено неверное предсказание ветвления
  - Ожидание результата предыдущей инструкции

https://www.technewstoday.com/how-to-check-cpu-cores-and-threads/

Переподписка на уровне процессора – технология Hyper Threading

SMT - simultaneous multithreading



Плюсы:

- Повышение производительности на 15 - 30% в некоторых задачах
- Увеличение числа пользователей обслуживаемых сервером
- Уменьшение времени отклика

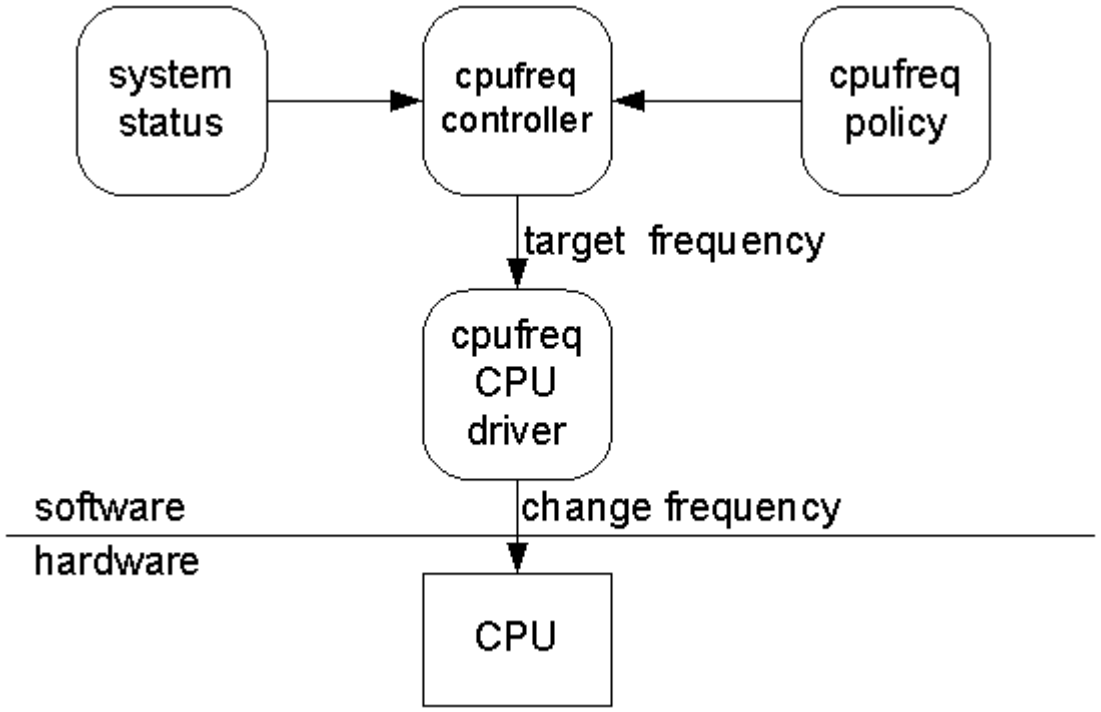
Минусы:

- В некоторых случаях, возможно снижение производительности
- Повышение фрагментации памяти

https://en.wikipedia.org/wiki/Hyper-threading

Управление частотой процессора CPUFreq Governors

Clock scaling allows you to change the clock speed of the CPUs on the fly.
This is a nice method to save battery power, because the lower the clock speed, the less power the CPU consumes.



Режимы работы CPUFreqGovernors

- Powersave
- Performance
- Userspace
- Ondemand

https://wiki.xenproject.org/wiki/Xen_power_management
https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt

Управление частотой процессора CPUFreq Governors

Возможно ли разная частота ядер процессора

- set cpufreq governor

```
# xenpm set-scaling-governor ondemand|performance|powersave
```

- get cpufreq P-state status

```
# xenpm get-cpufreq-states
cpu id        : 0
total P-states    : 4
usable P-states    : 4
current frequency  : 800 MHz
P0          : freq       [2534 MHz]
            transition [0000000000000000003]
            residency  [0000000000000002668 ms]
P1          : freq       [2533 MHz]
            transition [0000000000000000000]
            residency  [0000000000000000000 ms]
P2          : freq       [1600 MHz]
            transition [0000000000000000000]
            residency  [0000000000000000000 ms]
*P3         : freq       [0800 MHz]
            transition [0000000000000000003]
            residency  [0000000000000000237 ms]
```

- get cpufreq parameters

```
cpu id         : 0
affected_cpus      : *0 1
cpuinfo frequency   : max [2534000] min [800000] cur [800000]
scaling_driver     : acpi-cpufreq
scaling_avail_gov    : userspace performance powersave ondemand
current_governor    : ondemand
  ondemand specific :
    sampling_rate  : max [10000000] min [10000] cur [20000]
    up_threshold  : 80
scaling_avail_freq   : 2534000 2533000 1600000 *800000
scaling frequency   : max [2534000] min [800000] cur [800000]
```

https://wiki.xenproject.org/wiki/Xen_power_management

Доклад HighLoad++ 2017 настройка k8s: tips and tricks

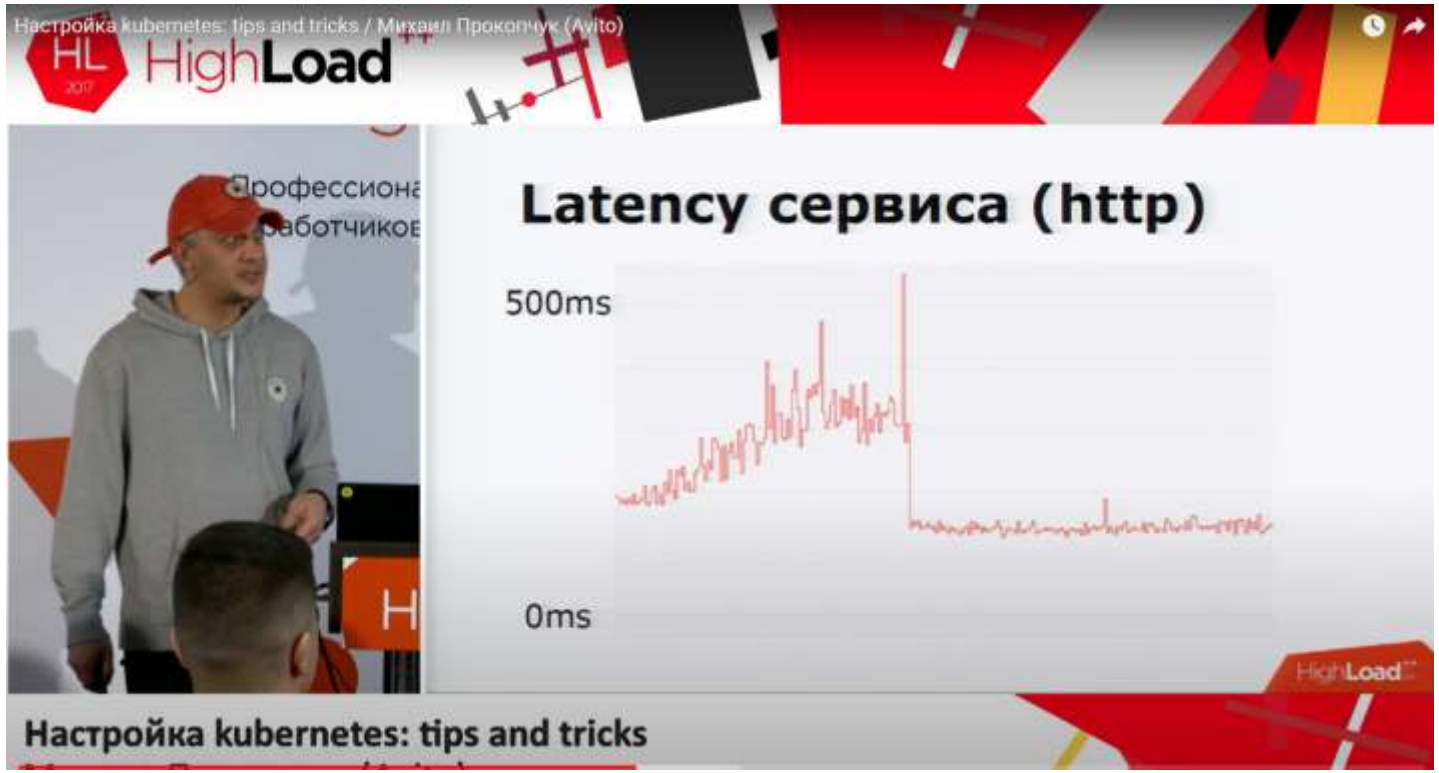
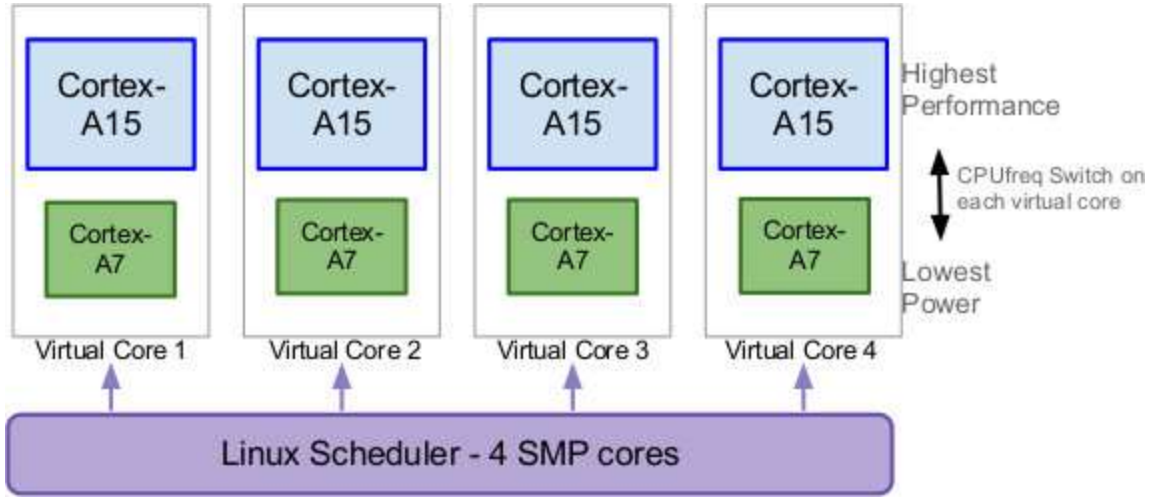


График за 36 часов для конкретного сервиса

https://www.youtube.com/watch?v=Zbe8k0cQpYo

Гибридная технология ARM big.LITTLE

Более эффективная технология энергопотребления, пришедшая из мобильный устройств



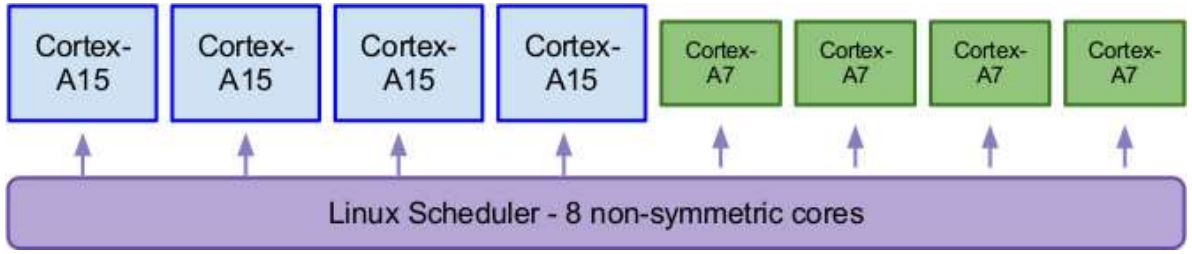
Принцип работы **(In-Kernel switcher - IKS)** :

- Пара ядер *big.LITTLE* работает как одно объединенное виртуальное ядро, при этом одновременно включено только одно
- *big* ядро используется в случае, когда нагрузка высокая
- *LITTLE* ядро, когда нагрузка низкая

https://en.wikipedia.org/wiki/ARM_big.LITTLE

Гибридная технология ARM big.LITTLE

Более эффективная технология энергопотребления, пришедшая из мобильный устройств



Принцип работы **(Heterogeneous multi-processing - HMP)** :

- В технологии HMP - одновременно работают, как *big* так и *LITTLE* ядра
- Потоки с высоким приоритетом или высокой интенсивностью вычислений отдаются *big* ядрам
- Низкоприоритетные или фоновые потоки отдаются *LITTLE* ядрам

https://en.wikipedia.org/wiki/ARM_big.LITTLE

Гибридная технология Intel Alder Lake

Более эффективная технология энергопотребления, пришедшая из мобильный устройств

Alder Lake-U (15W) [править | править код]

Серия	Модель	Ядра (Потоки)		Частота ЦП			
				Базовая		Max Turbo	
		P-ядро	E-ядро	P-ядро	E-ядро	P-ядро	E-ядро
Core i7	1265U	2 (4)	8 (8)	1.8 GHz	1.3 GHz	4.8 GHz	3.6 GHz
	1255U			1.7 GHz	1.2 GHz	4.7 GHz	3.5 GHz
Core i5	1245U			1.6 GHz		4.4 GHz	3.3 GHz
	1235U			1.3 GHz	0.9 GHz		
Core i3	1215U		4 (4)	1.2 GHz			
Pentium	8505	1 (2)					
Celeron	7305			1.1 GHz		-	-

Обозначения

- P-Core, Performance Core, “большие” ядра
- E-Core, Efficient Core, “маленькие” ядра

https://ru.wikipedia.org/wiki/Alder_Lake

Гибридная технология Intel Alder Lake + CPUFreq Governors

Приложение HWMonitor 1.52.0

sor	Value	Min	Max
Clocks			
P-Core #0	1995 MHz	798 MHz	4690 MHz
P-Core #1	4090 MHz	399 MHz	4690 MHz
E-Core #2	1297 MHz	798 MHz	3492 MHz
E-Core #3	3192 MHz	798 MHz	3492 MHz
E-Core #4	2095 MHz	798 MHz	3492 MHz
E-Core #5	2095 MHz	798 MHz	3492 MHz
E-Core #6	1895 MHz	798 MHz	3492 MHz
E-Core #7	1895 MHz	698 MHz	3492 MHz
E-Core #8	1297 MHz	898 MHz	3492 MHz
E-Core #9	3192 MHz	798 MHz	3492 MHz

	Value	Min	Max
Utilization			
Processor	12.0 %	0.0 %	27.4 %
P-Cores	27.6 %	0.0 %	46.7 %
CPU #0	26.2 %	0.0 %	81.3 %
CPU #1	28.4 %	0.0 %	83.1 %
CPU #2	35.4 %	0.0 %	84.4 %
CPU #3	0.0 %	0.0 %	84.6 %
E-Cores	4.2 %	0.0 %	24.0 %
CPU #4	3.1 %	0.0 %	36.9 %
CPU #5	6.2 %	0.0 %	34.4 %
CPU #6	3.1 %	0.0 %	40.0 %
CPU #7	1.5 %	0.0 %	30.8 %
CPU #8	1.5 %	0.0 %	29.2 %
CPU #9	4.6 %	0.0 %	32.3 %
CPU #10	1.5 %	0.0 %	43.1 %
CPU #11	1.5 %	0.0 %	57.8 %

https://www.cpuid.com/softwares/hwmonitor.html

Выводы - hardware (1/3)

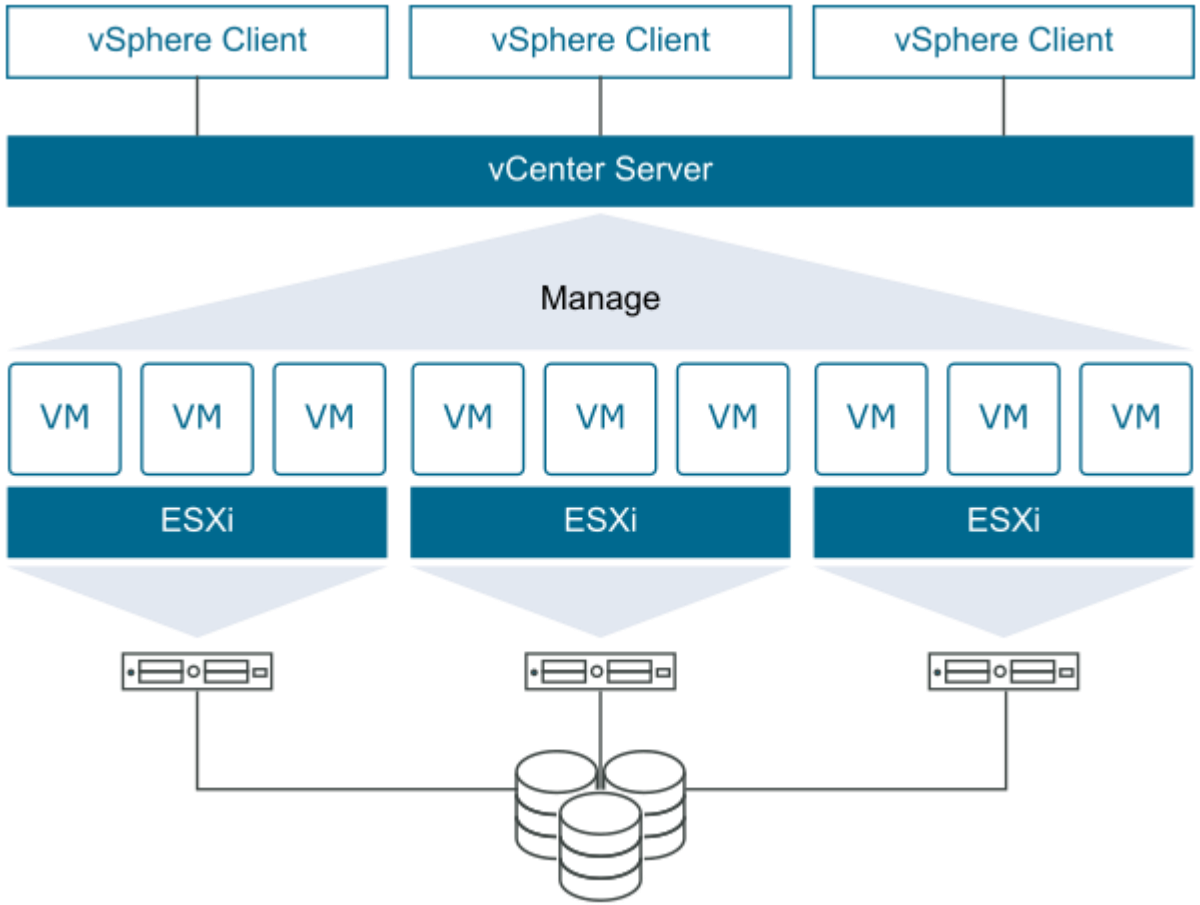
- При оценке производительности приложения, необходимо обращать внимание на:
  - Кол-во ядер процессора и их тактовую частоту
    - Потребуется при расчёте перепродажи на уровне виртуализации
  - Режим *CPUFreq Governors* на уровне операционной системы или виртуализации
    - Особенно заметно, когда нагрузка значительно меняется в конкретный момент времени
  - Диагностировать проблемы с производительностью в порядке: *Диски > Процессор > Оперативная память > Сеть*

MONOPOLY.
ONLINE

Виртуализация

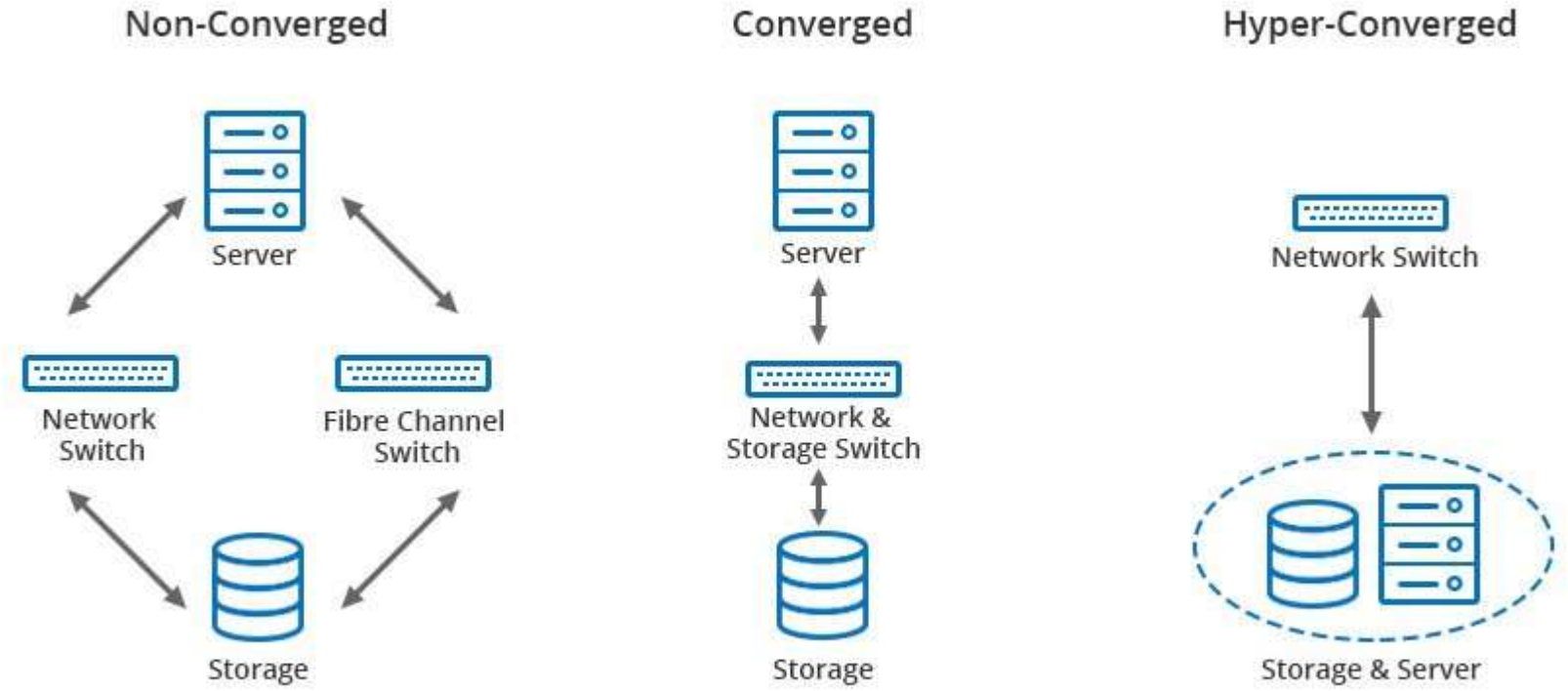
Конвергентная и гиперконвергентая инфраструктура

Когда вычислительные мощности и хранилища связаны через сеть



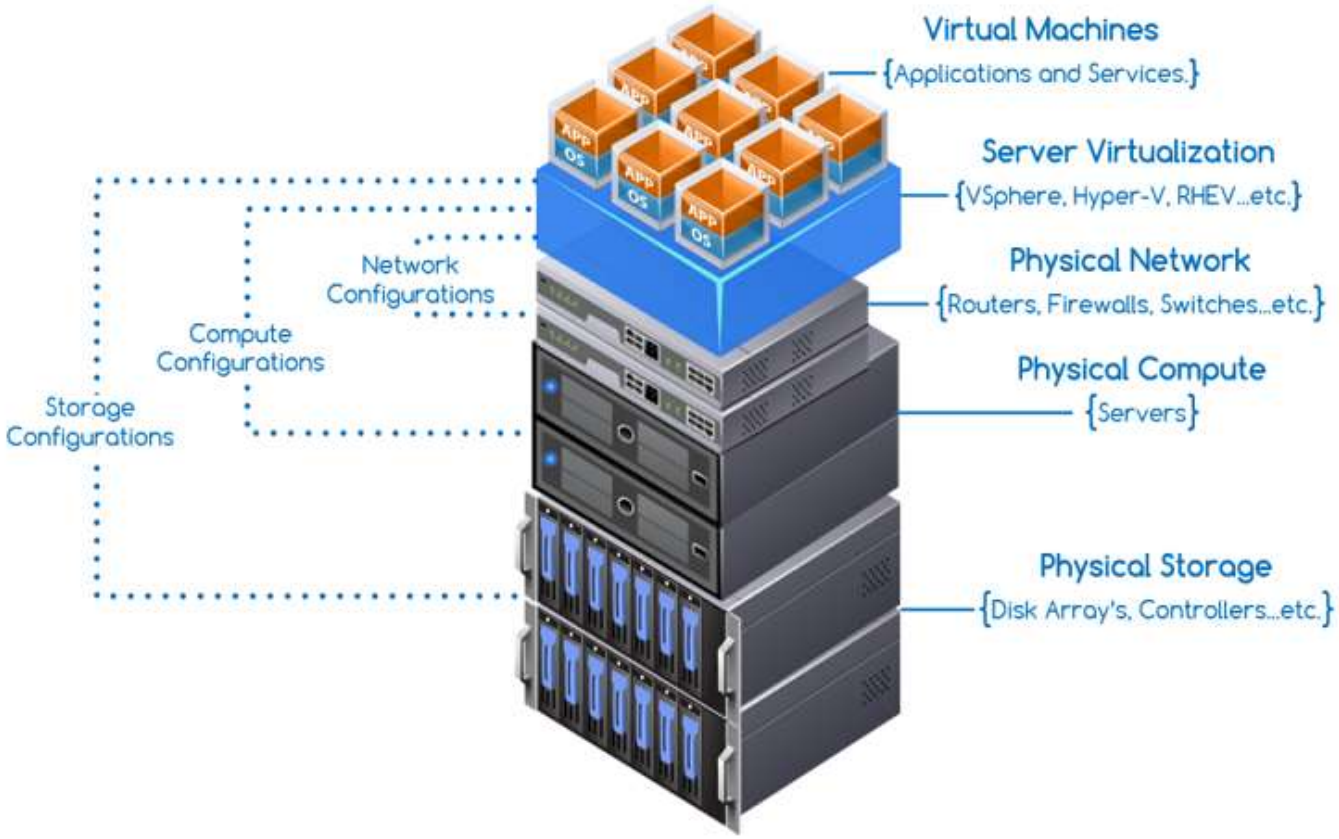
https://docs.vmware.com/en/VMware-vSphere/index.html

Традиционная, конвергентная и гиперконвергентая инфраструктура



https://en.wikipedia.org/wiki/Converged_infrastructure

Конвергентная инфраструктура



Плюсы

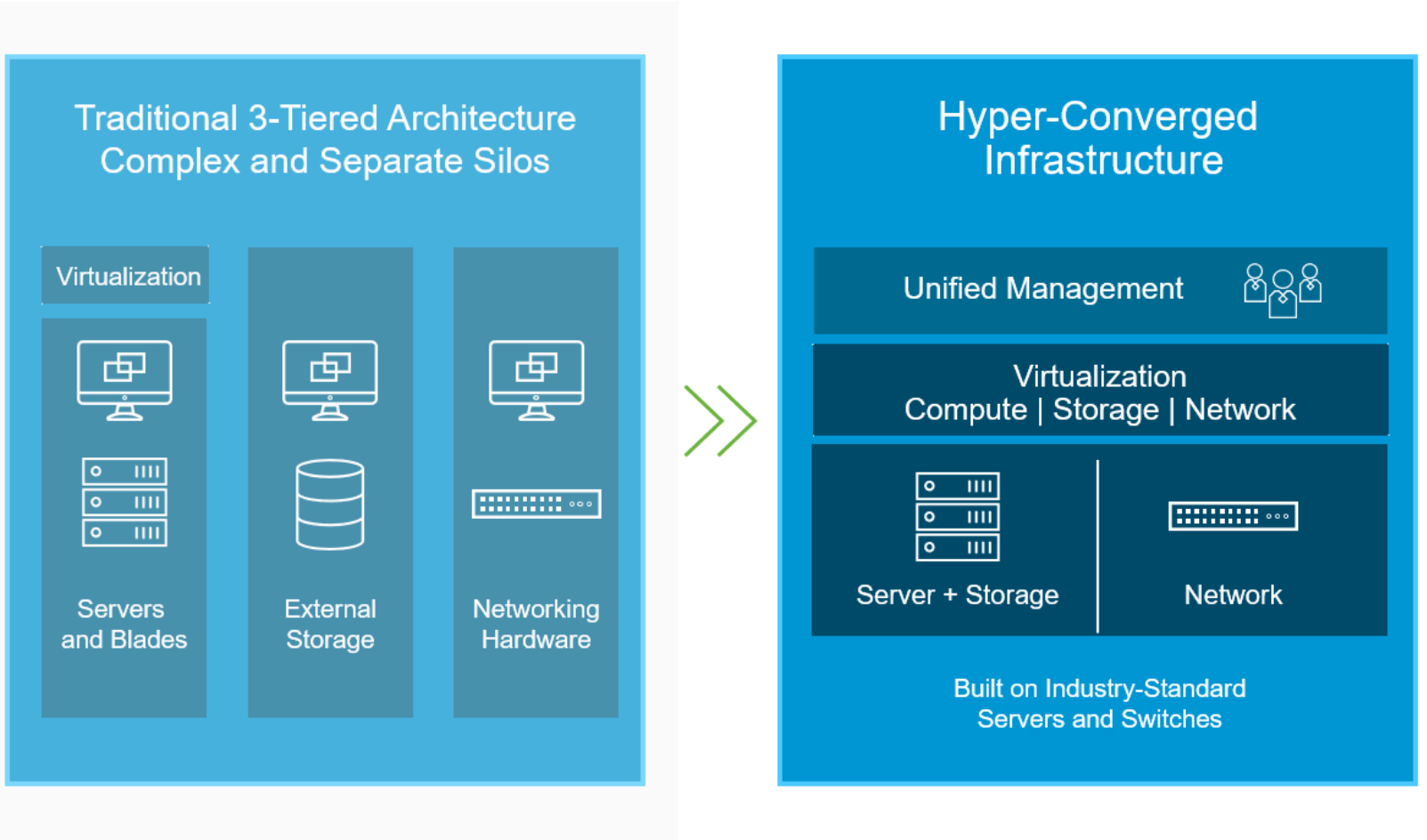
- Гибкое масштабирование каждого из узлов инфраструктуры
- Стандартизация и упрощение процесса обслуживания узлов инфраструктуры

Минусы

- Дорогие решения привязанные к конкретным вендорам оборудования
- Необходимость значительных капитальных затрат на старте проекта

https://itglobal.com/ru-ru/company/blog/converged-hyperconverged-environments/

Эволюция конвергентной в гиперконвергентную инфраструктуру



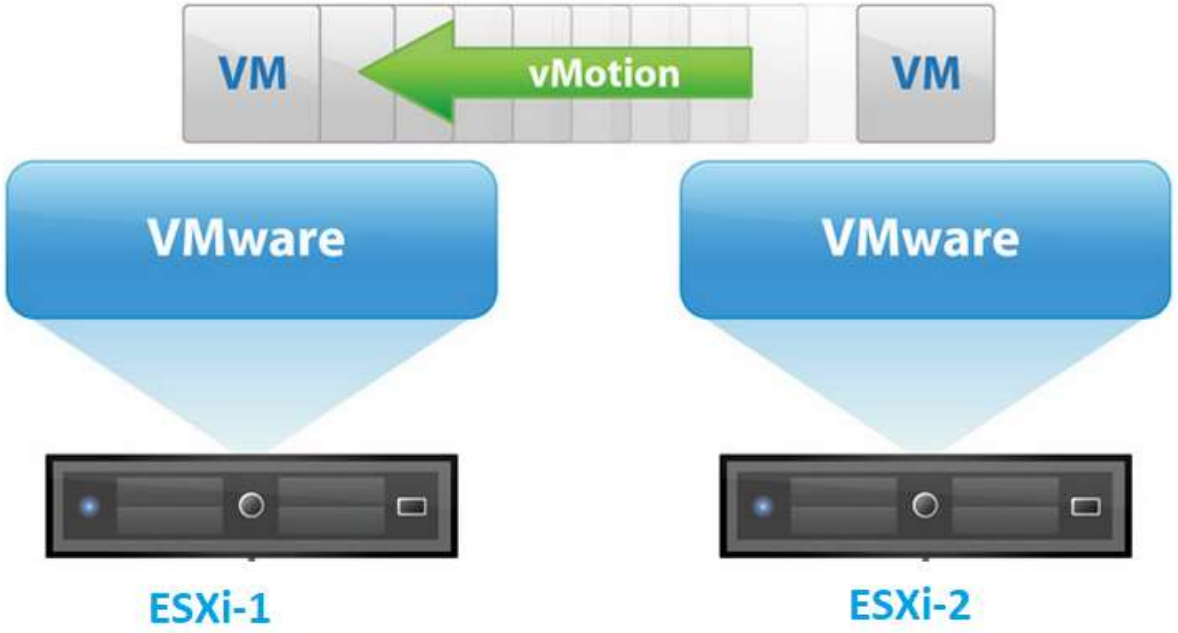
https://masteksystems.com/solutions/hyper-converged-infrastructure

MONOPOLY.
ONLINE

Виртуализация

Технологии vMotion и DRS (Dynamic Resource Scheduling)

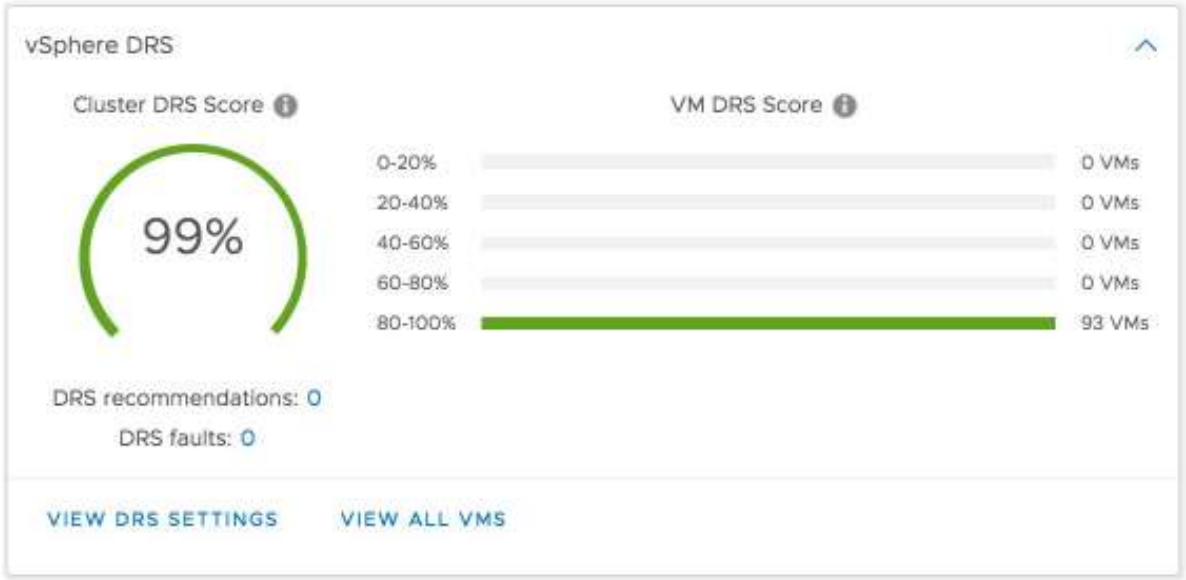
Технология VMware vMotion



- Обслуживание оборудования без простоев (downtime) и прерывание бизнес операций
- Перемещение виртуальных машин с упавших или “тормозящих” серверов
- Более эффективное использование ресурсного пула

https://www.vmware.com/products/vsphere/vmotion.html

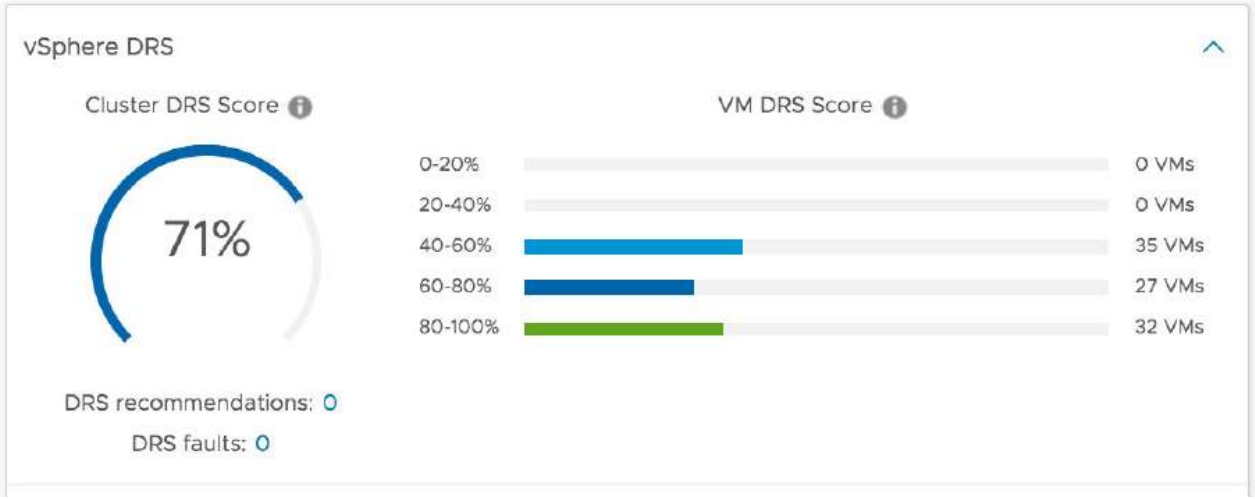
Балансирование производительности средствами DRS



Name	State	Consumed CPU %	Consumed Memory %
10.196.6.68	Connected	29%	30%
10.196.6.69	Connected	29%	30%
10.196.6.70	Connected	29%	30%
10.196.6.71	Connected	29%	30%

https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/performance/drs-vsphere7-perf.pdf

Балансирование производительности средствами DRS

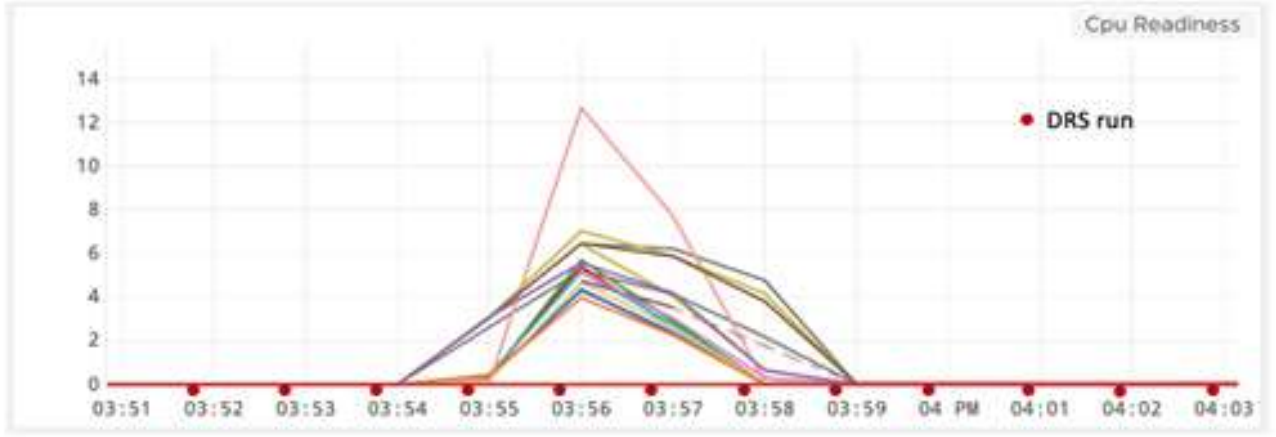


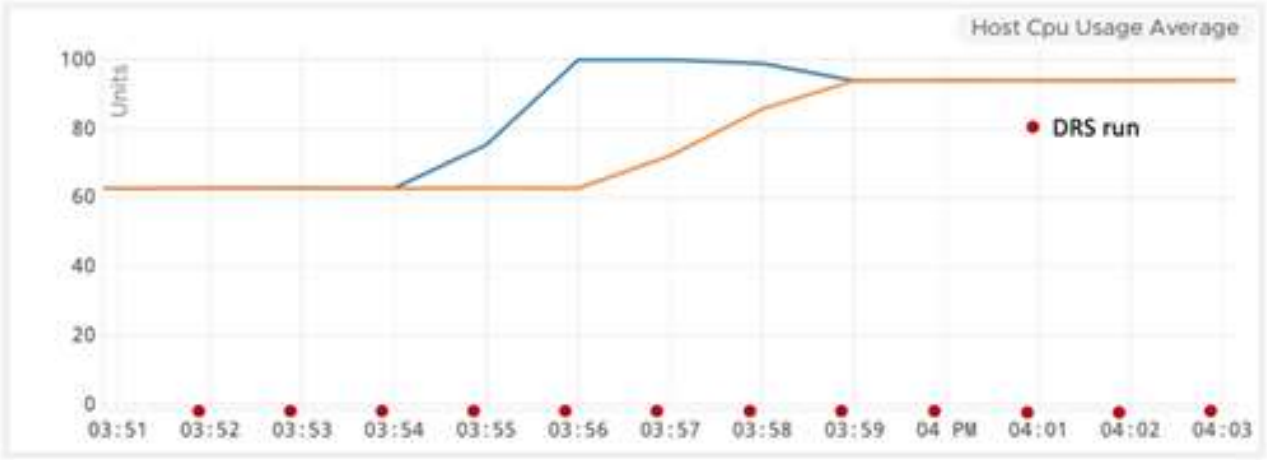
Name	State	Consumed CPU %	Consumed Memory %
10.196.6.68	Connected	99%	59%
10.196.6.69	Connected	73%	71%
10.196.6.70	Connected	29%	30%
10.196.6.71	Connected	29%	30%

https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/performance/drs-vsphere7-perf.pdf

Технологии vMotion и DRS (Dynamic Resource Scheduling)

Балансирование производительности средствами DRS

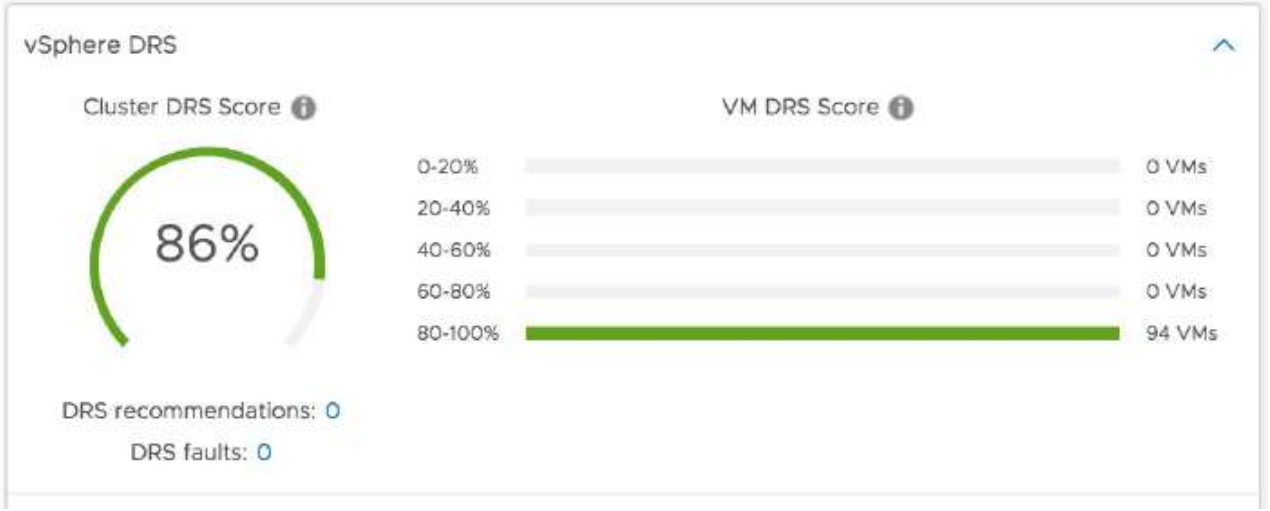




https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/performance/drs-vsphere7-perf.pdf

Технологии vMotion и DRS (Dynamic Resource Scheduling)

Балансирование производительности средствами DRS

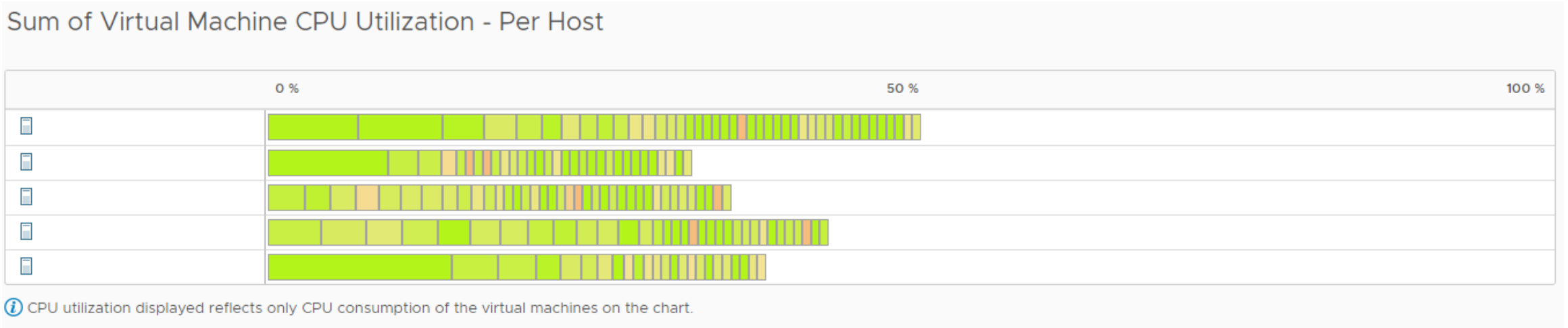


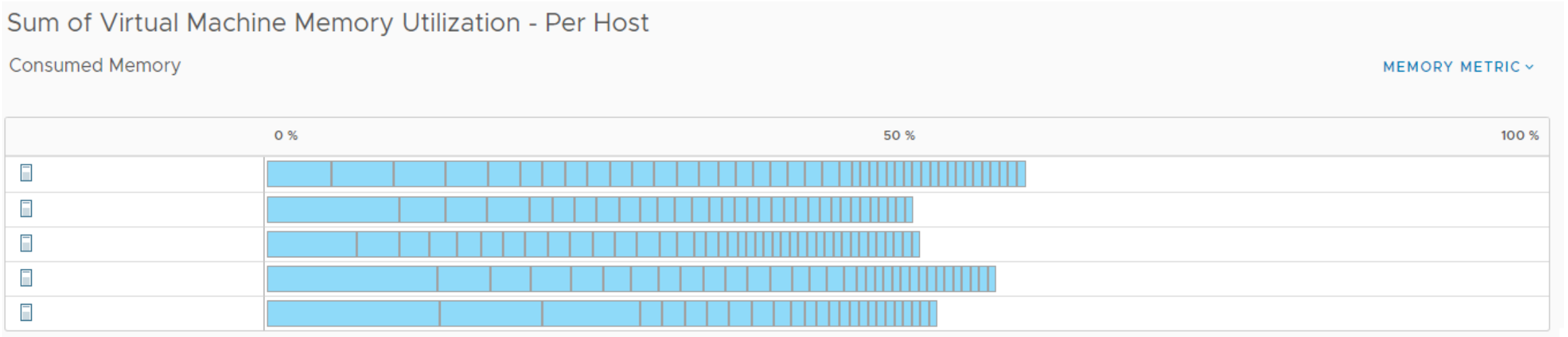
Name	State	Consumed CPU %	Consumed Memory %
10.196.6.68	Connected	73%	41%
10.196.6.69	Connected	58%	59%
10.196.6.70	Connected	66%	43%
10.196.6.71	Connected	65%	47%

https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/performance/drs-vsphere7-perf.pdf

Технологии vMotion и DRS (Dynamic Resource Scheduling)

Стандартизация и минимизация размеров виртуальных машин





Стандартизация и минимизация размеров виртуальных машин

Размер инстанса	Виртуальные ЦПУ	Память (ГиБ)	Хранилище инстансов (ГБ)	Пропускная способность сети (Гбит/с)***	Пропускная способность EBS (Мбит/с)
c6g.medium	1	2	Только EBS	До 10	До 4750
c6g.large	2	4	Только EBS	До 10	До 4750
c6g.xlarge	4	8	Только EBS	До 10	До 4750
c6g.2xlarge	8	16	Только EBS	До 10	До 4750
c6g.4xlarge	16	32	Только EBS	До 10	4750
c6g.8xlarge	32	64	Только EBS	12	9000
c6g.12xlarge	48	96	Только EBS	20	13 500
c6g.16xlarge	64	128	Только EBS	25	19 000

Мы видим как vCPU и RAM растет x2 в рамках конкретной линейки инстансов c6g

https://aws.amazon.com/ru/ec2/instance-types/

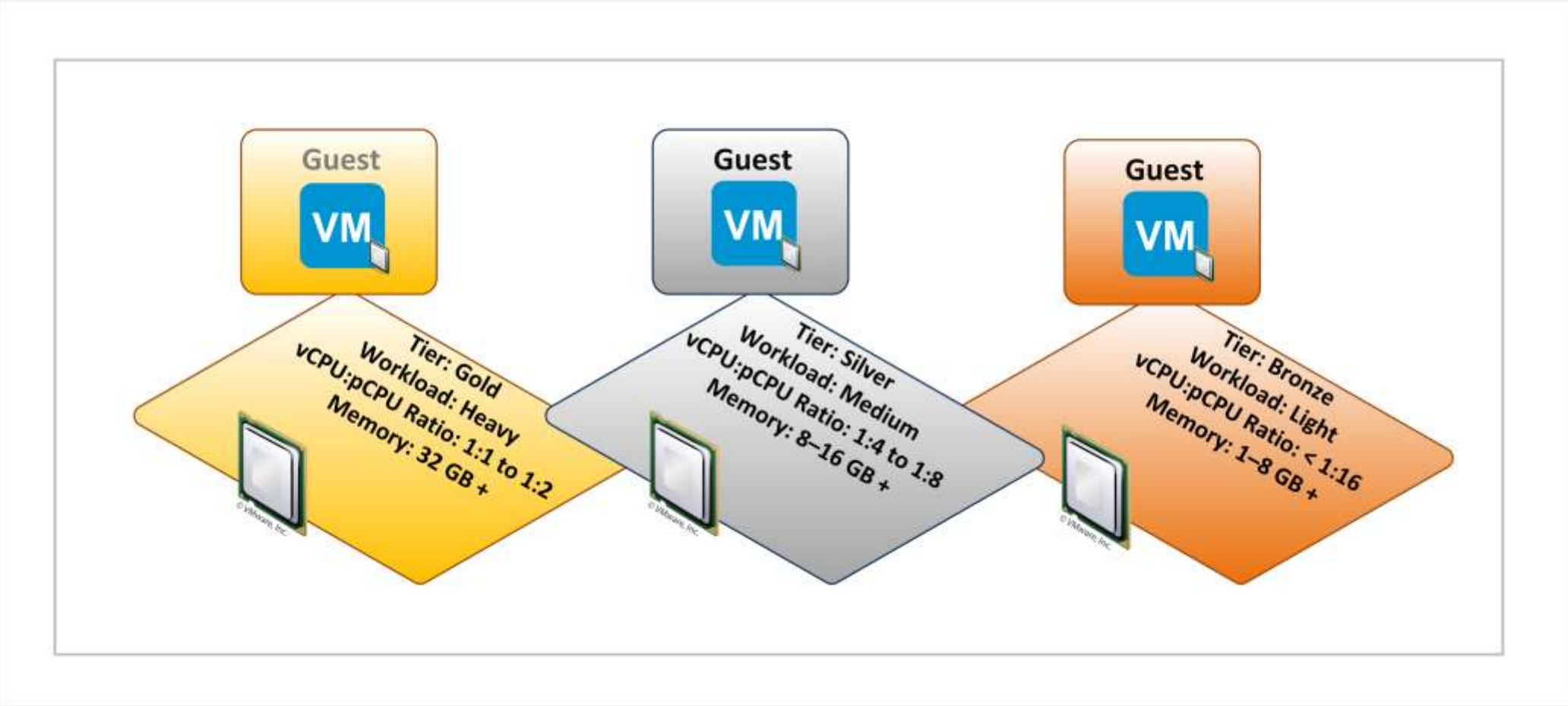
MONOPOLY.
ONLINE

Виртуализация

Переподписка вычислительных ресурсов

Переподписка вычислительных ресурсов

Допустимые коэффициенты переподписки для кластера vCPU/CPU



https://download3.vmware.com/vcat/vmw-vcloud-architecture-toolkit-spv1-webworks/index.html

Переподписка вычислительных ресурсов

В технической документации Dell: “Demystifying CPU Ready (% RDY) as a Performance Metric” рекомендуется:

vCPU / CPU	vCPU / CPU	Комментарий
1:1	3:1	Проблем с производительностью не будет
3:1	5:1	Могут возникать случаи с деградацией производительности
6:1	более	Часто возникают проблемы с производительностью

В дополнение следить за метрикой CPU Ready, чтобы она не превышала 5 %

https://communities.vmware.com/t5/VMware-vSphere-Documents/Dell-Best-Practices-for-Oversubscription-of-CPU-Memory-and/ta-p/2790280?attachment-id=92833

Переподписка вычислительных ресурсов

Реальный мир

vCPU / CPU	vCPU / CPU	Комментарий
10 : 1	15 : 1	Реальные коэффициенты переподписки

Рекомендация из документа Dell “Demystifying CPU Ready (% RDY) as a Performance Metric”:

vCPU / CPU	vCPU / CPU	Комментарий
6 : 1	8 : 1	Стремиться к интервалу

https://communities.vmware.com/t5/VMware-vSphere-Documents/Dell-Best-Practices-for-Oversubscription-of-CPU-Memory-and/ta-p/2790280?attachment-id=92833

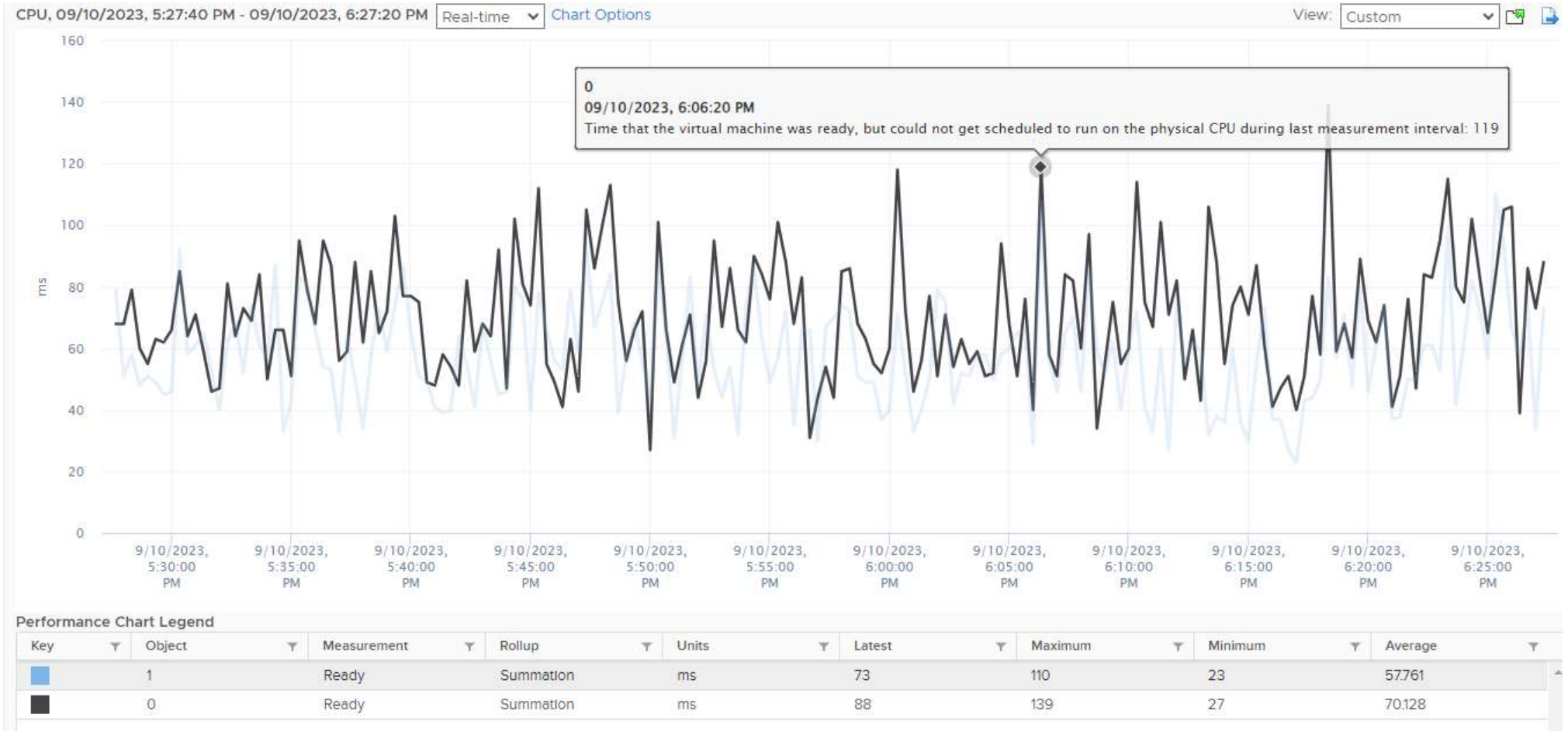
MONOPOLY.
ONLINE

Виртуализация

Метрики производительности: RDY, WAIT, CSTP

Метрики производительности WAIT, RDY, CSTP

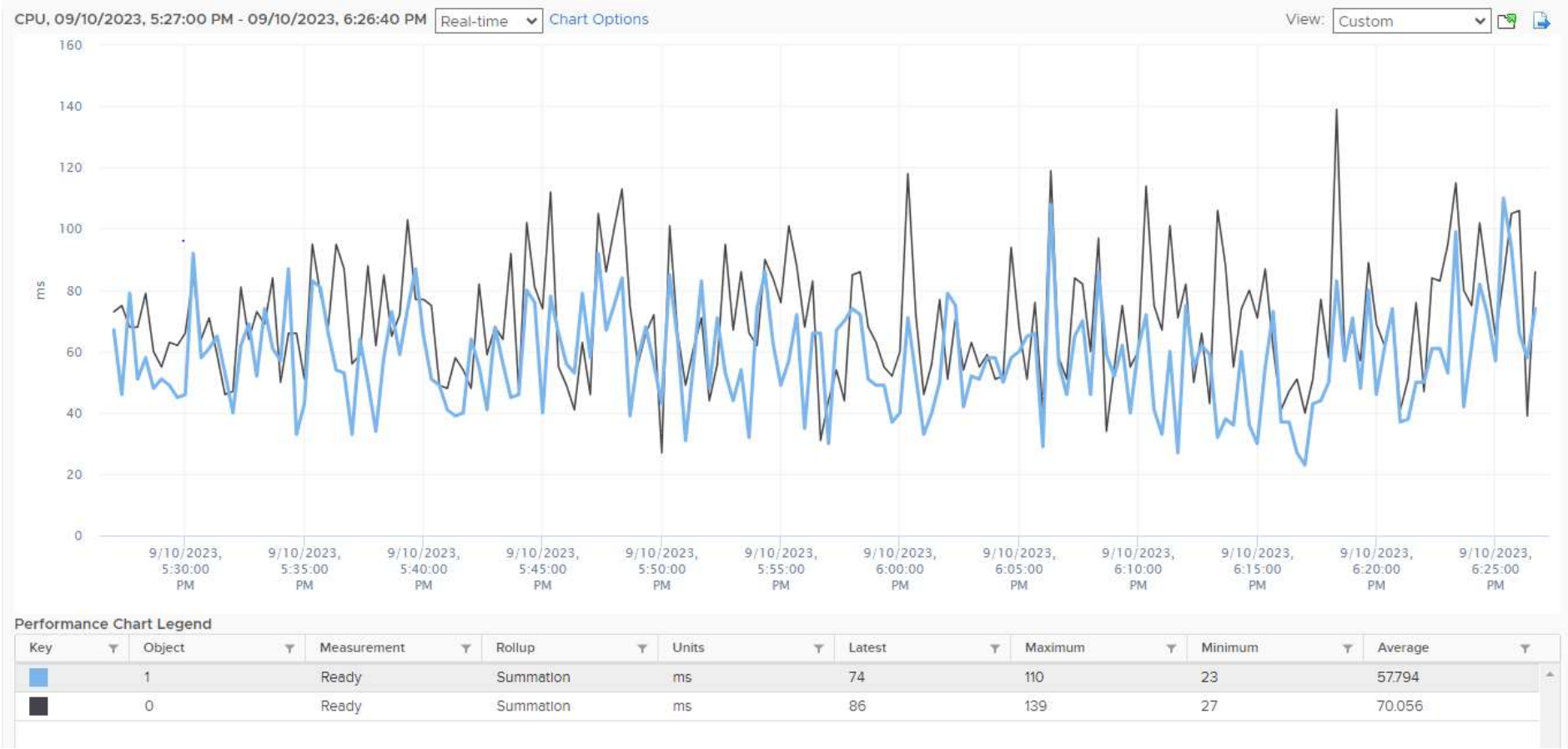
- Wait (WAIT) - % времени или милисекунды, течение которого ВМ ждёт окончания какой-то активности VmKernel.
- Причина: чаще всего это IO-активность: дисковая или сетевая



Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum	Average
	1	Ready	Summation	ms	73	110	23	57.761
	0	Ready	Summation	ms	88	139	27	70.128

Метрики производительности WAIT, RDY, CSTP

- CPU Ready (RDY) - % времени или милисекунды, когда VM готова считать, но физические процессоры заняты другими процессами (системными или другими VM).



Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum	Average
	1	Ready	Summation	ms	74	110	23	57.794
	0	Ready	Summation	ms	86	139	27	70.056

Метрики производительности WAIT, RDY, CSTP

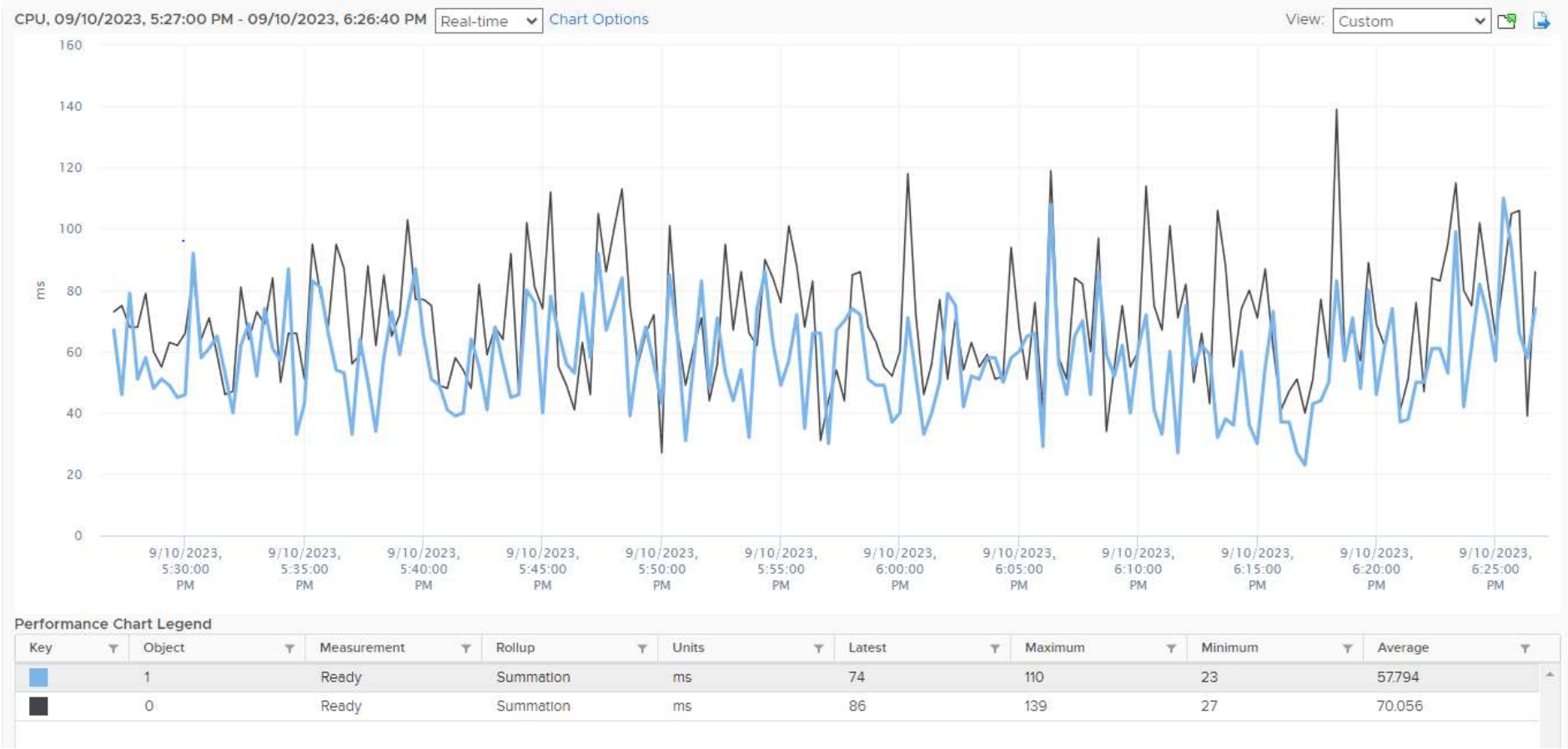
Причины высокого CPU Ready (RDY):

- Большая переподписка по процессору – (vCPU / CPU > 10)
- Большое кол-во VM интенсивно потребляют процессор
- Наличие Oversized VM (виртуальная машина с большим кол-во недозагруженных ядер)

  Планировщик будет тормозить машину, пока не найдет необходимое кол-во ядер (метрика Co-Stop)
  - Лучше 2 vCPU с 80%, чем 8 vCPU с 20%
  - Мы лишаемся оптимизации NUMA
- Мы уперлись в лимиты Resource Pool
  - В этом случае будет увеличиваться значение Max-Limited (%MLMTD)

Метрики производительности WAIT, RDY, CSTP

- CPU Ready (RDY) - % времени или милисекунды, когда VM готова считать, но физические процессоры заняты другими процессами (системными или другими VM).



Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum	Average
	1	Ready	Summation	ms	74	110	23	57.794
	0	Ready	Summation	ms	86	139	27	70.056

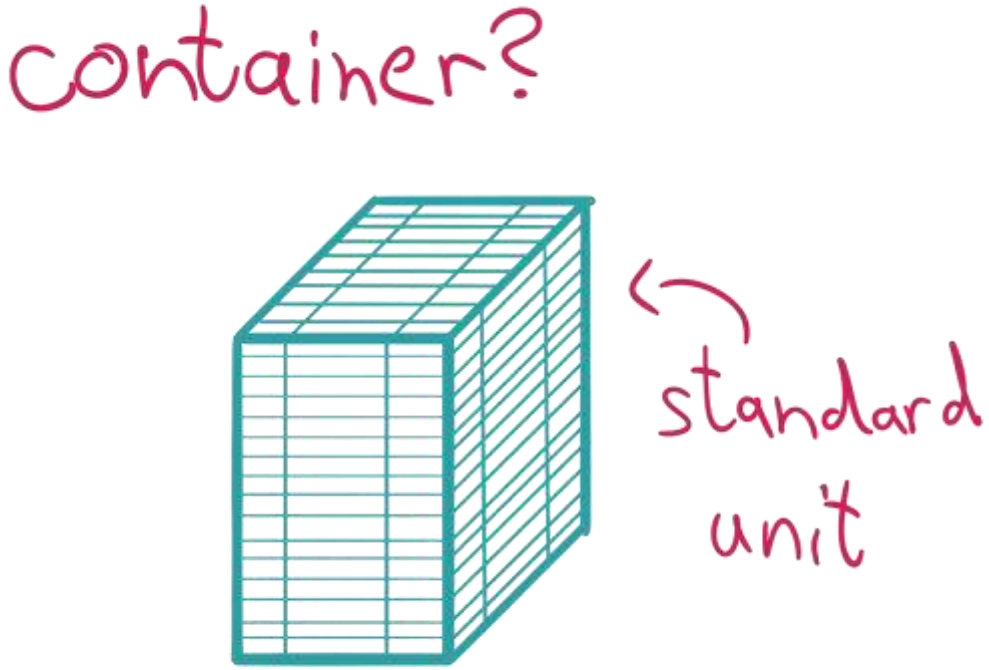
Выводы и рекомендации - VMs (2/3)

- Проанализируйте метрики %USED, %RDY, %WAIT, %CSTP по самым загруженным виртуальным машинам
- Снижайте количество Oversized VM
  - Лучше 2 vCPU с 80%, чем 8 vCPU с 20%
- Снижайте коэффициенты переподписки до vCPU / CPU < 10
  - Чем больше переподписка, тем меньше должны быть VM
- Стандартизируйте размеры VM
  - Облегчите работу DRS и равномерно загрузите кластер
- Зарезервируйте ресурсы под ключевые VM: базы данных, кластера Kafka или RabbitMq
  - Удобно назначать ресурсы не на каждую машину, а использовать Reservation Pool

MONOPOLY.
ONLINE

Контейнеризация

Контейнеризация как следующий уровень абстракции в инфраструктуре



Преимущества контейнеризации:

- Простота и стандартизация
  - Развертывании приложений
  - Масштабировании приложений
    - Как горизонтальное так и вертикальное
- Одинаково работает в инфраструктуре облачного провайдера, так и на собственном железе
- Контейнеризация довольно дешевая в сравнении с виртуализацией

https://github.com/saschagrunert/demystifying-containers/blob/master/part1-kernel-space/post.md

Виртуализация никуда не делась

В терминах k8s VMs называются нодами

Pool: prd-kubeworker-						26 Nodes
Active	prd-kubeworker-1 10.22.1.112	Worker	v1.17.14 19.3.11	3.8/7 Cores	10.6/14.5 GiB	17/40
Active	prd-kubeworker-2 10.22.1.119	Worker	v1.17.14 19.3.11	3.4/7 Cores	11.3/14.5 GiB	21/40
Active	prd-kubeworker-3 10.22.1.104	Worker	v1.17.14 19.3.11	4.9/7 Cores	12.6/14.5 GiB	17/40

Ноды production - кластера

Pool: stg-kubeworker-						5 Nodes
Active	stg-kubeworker-1 10.22.5.209	Worker	v1.17.14 19.3.11	1.7/7 Cores	9.7/38.1 GiB	85/110
Active	stg-kubeworker-2 10.22.2.44	Worker	v1.17.14 19.3.11	2.2/7 Cores	9/38.1 GiB	86/110
Active	stg-kubeworker-3 10.22.2.37	Worker	v1.17.14 19.3.11	2/7 Cores	10.1/38.1 GiB	92/110

Ноды staging - кластера

Однако есть нюансы при работе планировщика k8s

Он срабатывает только при деплое

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo
  namespace: mem-example
spec:
  containers:
  - name: memory-demo-ctr
    image: polinux/stress
    resources:
      requests:
        memory: "100Mi"
      limits:
        memory: "200Mi"
    command: ["stress"]
    args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: cpu-demo
  namespace: cpu-example
spec:
  containers:
  - name: cpu-demo-ctr
    image: vish/stress
    resources:
      limits:
        cpu: "1"
      requests:
        cpu: "0.5"
    args:
    - -cpus
    - "2"
```

- https://kubernetes.io/docs/tasks/configure-pod-container/assign-cpu-resource/
- https://kubernetes.io/docs/tasks/configure-pod-container/assign-memory-resource/

Укажем request по CPU превышающий размер ноды

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: cpu-demo-2
  namespace: cpu-example
spec:
  containers:
  - name: cpu-demo-ctr-2
    image: vish/stress
    resources:
      limits:
        cpu: "100"
      requests:
        cpu: "100"
    args:
    - -cpus
    - "2"
```

```
kubectl apply -f https://k8s.io/examples/pods/resource/cpu-request-limit-2.yaml
```

```
kubectl describe pod cpu-demo-2 --namespace=cpu-example
```

```
Events:
  Reason                        Message
  ------                        -------
  FailedScheduling      No nodes are available that match all of the following predicates:: Ins
```

- https://kubernetes.io/docs/tasks/configure-pod-container/assign-cpu-resource/

Укажем request по памяти превышающий размер ноды

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo-3
  namespace: mem-example
spec:
  containers:
  - name: memory-demo-3-ctr
    image: polinux/stress
    resources:
      requests:
        memory: "1000Gi"
      limits:
        memory: "1000Gi"
    command: ["stress"]
    args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]
```

```
kubectl apply -f https://k8s.io/examples/pods/resource/memory-request-limit-3.yaml
```

```
kubectl describe pod memory-demo-3 --namespace=mem-example
```

```
Events:
  ...  Reason            Message
       ------            -------
  ...  FailedScheduling  No nodes are available that match all of the following predicates:
```

- https://kubernetes.io/docs/tasks/configure-pod-container/assign-memory-resource/

Выводы и рекомендации – k8s (3/3)

- Установить параметры:
  - Request CPU, RAM равный текущей использованию +25%
  - Limit CPU, RAM равный Request x2 с выравниванием в большую сторону
    - Если ресурсов много, то установить Limit только для самых прожорливых сообщений

MONOPOLY.
ONLINE

Спасибо за внимание!

Станислав Флусов

- Telegram: https://t.me/sflusov