New scheduling algorithm for Kubernetes

Albert Ludwigs University Freiburg, Seminar Algorithms for Computer Networks

About this presentation

Presenter: Jürgen Mattheis, BSc. Student (Computer Science)
Supervisor: Dr. Prof. Christian Schindelhauer
Date: 01/06/2021
Main Source: [1] by José Santos, Tim Wauters, Bruno Volckaert and Filip DeTurck
New scheduling algorithm for Kubernetes

Albert Ludwigs University Freiburg, Seminar Algorithms for Computer Networks

About this presentation

Presenter: Jürgen Mattheis, BSc. Student (Computer Science)
Supervisor: Dr. Prof. Christian Schindelhauer
Date: 01/06/2021
Main Source: [1] by Jos´e Santos, Tim Wauters, Bruno Volckaert and Filip DeTurck
Motivation

Kubernetes:

- orchestrator
- automates the management of containers [2]
Motivation

Kubernetes:

- orchestrator
- automates the management of containers [2]

sandbox for a microservice

[kub]: Kubernetes Logo

Page: 04/74
Motivation

Service:

• set of software functionalities used by clients
Motivation

Microservice:

- independent service with small set of functionalities
- communicates over network

Password: **********
Motivation

Cloud Computing:

Password: **********
Motivation

Fog Computing:

Password: ***********
Fog Computing:

Container (within a Pod)
Motivation

Fog Computing:

[fog]: Fog
[clo]: Clouds
[par]: water particles

fog ↔ clouds
Motivation

Smart Cities:

- smart lighting
- traffic monitored in real time
- began in 2009
- ≈170 projects

[sma]: Amsterdam smart city initiative
Motivation

Smart Cities:

- smart lighting
- traffic monitored in real time
- began in 2009
- ≈170 projects

[smart city initiative: Amsterdam]
Motivation

Fog Computing:
Motivation

Fog Computing:

Network Latency:

RTT

Kubernetes
Motivation

Fog Computing:

Network Latency: RTT

Bandwidth:

OR

Page: 15/74
Motivation

Fog Computing:

Kube-scheduler (KS) vs. Network-Aware Scheduler (NAS)

Kubernetes
Introduction

Virtual Machines:

- Operating System (OS) **virtualized** inside another OS  [9]
Introduction

Virtual Machines:

- Operating System (OS) **virtualized** inside another OS  [9]
Introduction

Container:

- Sandbox for a **Microservice** to run in
Container:

- Sandbox for a **Microservice** to run in

network latency, load balancing, testing

Password: ***********
Introduction

Container:

- **container runtime**: software responsible for running containers [8]
Introduction

Kubernetes:
Introduction

Kubernetes:

- orchestrator
- manages **containers**
- greek word for helmsman
- abbreviation K8s

[kub]: Kubernetes timeline

[dna]: Shared DNA

Brog (proprietary)
Omega (proprietary)
Kubernetes (open-source)

2014
2021

now
Introduction

Kubernetes:

- **Node**: Represents a compute resource in Kubernetes.
- **Pod**: A collection of one or more containers that run as a single unit.
- **Master**: The control plane of the Kubernetes cluster.
- **Container**: An isolated process or group of processes运行于一个或多个容器中运行。
Introduction

Kubernetes:

- Master
  - API Server
  - control manager
  - scheduler
  - cluster store

- manifest file

Page: 26/74
Introduction

Kubernetes:

- Master
  - API Server
  - cluster store
  - control manager
  - scheduler

manifest file

Page: 27/74
Kubernetes:

Node

- network proxy
- kubelet
- container runtime

Pod Pod Pod Pod Pod
Introduction

Kubernetes:

Node

network proxy
kubelet
container runtime

Pod
Pod
Pod
Pod
Pod
Pod
Introduction

Kubernetes:

Node:
- network proxy
- kubelet
- container runtime

Pod:

Pod

Pod

Pod

Pod

Pod

Pod
Kube-scheduler:

- Scheduler: Nodes $\rightarrow$ Queue: node $\rightarrow$ pod
Introduction

Kube-scheduler:

- Scheduler: Nodes $\rightarrow$ Queue: node $\rightarrow$ pod
- Queue:

```
Pod  Pod  Pod  Pod  Pod  Pod  Pod  Pod  Pod  Pod
```
**Introduction**

**Kube-scheduler:**

- **Scheduler**: Nodes $\rightarrow$ Queue: node $\rightarrow$ pod
- **Queue**: 
  - Pod
  - Pod
  - Pod
  - Pod
  - Pod
  - Pod
  - Pod
  - Pod
  - Pod
  - Pod
- **Nodes**: 
  - Node
  - Node
  - Node
  - Node
  - Node
  - Node
  - Node
  - Node
  - Node
  - Node

**Filtering**: 
- predicate 1
  - pod.attribute > node.value

<table>
<thead>
<tr>
<th>Pod</th>
<th>Pod</th>
<th>Pod</th>
<th>Pod</th>
<th>Pod</th>
<th>Pod</th>
<th>Pod</th>
<th>Pod</th>
<th>Pod</th>
<th>Pod</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node</td>
<td>Node</td>
<td>Node</td>
<td>Node</td>
<td>Node</td>
<td>Node</td>
<td>Node</td>
<td>Node</td>
<td>Node</td>
<td>Node</td>
</tr>
</tbody>
</table>
Introduction

Kube-scheduler:

- Scheduler : Nodes $\rightarrow$ Queue : node $\rightarrow$ pod
- Queue:

```
| Pod | Pod | Pod | Pod | Pod | Pod | Pod | Pod | Pod | Pod | Pod |
```

- Filtering: predicate 2
- Nodes:

```
```

$\text{pod.attribute} > \text{node.value}$
Kube-scheduler:

- Scheduler: Nodes → Queue: node → pod
- Queue:

```
Pod Pod Pod Pod Pod Pod Pod Pod Pod Pod
```

- Priority Calculation: priority 1
- Nodes:

```
Node Node Node Node Node Node Node Node Node Node
```

Page: 35/74
Introduction

Kube-scheduler:

- Scheduler: Nodes → Queue: node → pod
- Queue:
  - Pod
  - Pod
  - Pod
  - Pod
  - Pod
  - Pod
  - Pod
  - Pod
  - Pod
  - Pod

- Nodes:
  - Node
  - Node
  - Node
  - Node
  - Node
  - Node
  - Node
  - Node
  - Node
  - Node

Priority Calculation: priority 2

max({15, 14, 15, 16, 09})
# Introduction

**Kube-scheduler:**

<table>
<thead>
<tr>
<th>Labels</th>
<th>Taints</th>
<th>Affinity/Anti-Affinity rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pod (key, value)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node (key, value)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[3]
Kube-scheduler:

**Labels**
- Pod: (key, value)

**Taints**
- Node: (key, value, NoSchedule)
- Pod: tolerance for (key, value)
- Pod: no toleration for (key, value)

**Affinity/Anti-Affinity rules**
Kube-scheduler:

Labels

Pod: (key, value)
Node: (key, value)

Taints

Pod: (key, value, NoSchedule)
Node: (key, value, NoSchedule)
Pod: (key, value)
Pod: (key, value)

Affinity/Anti-Affinity rules

Pod
Pod
Pod
Pod
Node

- podAffinity
- podAntiAffinity
- nodeAffinity
Reason for the Network-Aware Scheduler (NAS):

- **Kube-scheduler (KS)** considers:
  - CPU usage rates
  - RAM usage rates
  - network latency usage rate
  - bandwidth usage rate
Reason for the Network-Aware Scheduler (NAS):

- **Kube-scheduler (KS)** considers:
  - CPU usage rates
  - RAM usage rates
  - network latency usage rate
  - bandwidth usage rate

[Network-Aware Scheduler (NAS)]

[Smart City]
Network-Aware Scheduler (NAS):

**Working Principle**

**Kube-scheduler**

**Filtering: Applying Predicates**

{ Remaining Nodes }

**Network-Aware Scheduler Extender**

**Filtering AND Priority Calculation:**

{ Remaining Nodes }

12  12
Network-Aware Scheduler (NAS):

Network-Aware Scheduler
Network-Aware Scheduler (NAS):

Node (RTT, 42)

Network-Aware Scheduler

Node (Bandwidth, 4.2)
Network-Aware Scheduler (NAS):

Working Principle

Network-Aware Scheduler

(Node, A, NoSchedule)
(RTT, 42)

(Node, D, NoSchedule)
(Bandwidth, 4.2)

Pod

Pod

Pod

Pod

no toleration for (Location, A)

toleration for (Location, A)

no toleration for (Location, A)

toleration for (Location, A)

toleration for (Location, D)

to toleration for (Location, D)

toleration for (Location, B)
Network-Aware Scheduler (NAS):

**Working Principle**

- **Node**
  - (Service, 3)
  - tolleration for (Location, A)
  - (Service, 4)
  - tolleration for (Location, A)

- **Pod**
  - (Location, A, NoSchedule)
  - (RTT, 42)

- **Network-Aware Scheduler**
  - uses
  - (Location, D, NoSchedule)
  - (Bandwidth, 4.2)

- **anti-affinity:** same process
  - (Service, 2)
  - tolleration for (Location, A)
  - no toleration for (Location, A)
  - (Service, 2)
  - tolleration for (Location, D)
  - (Service, 2)
  - tolleration for (Location, B)

- **Pod**
  - (Location, B)
NAS algorithm:

- initialization of variables, dealing with requests etc.
- **Input:** `nodes = {node1, node2, ...}, pod`
- **Output:** `selected_node`

**Variables:** `min_rtt = ∞, nodes_copy = nodes`

```plaintext
for node in nodes {
    rtt = getRTT(node, pod.location)
    min_rtt = getMin(min_rtt, rtt)
}
```
NAS algorithm:

- initialization of variables, dealing with requests etc.
- Input: , Pod
- Output: selected_node

\[
\text{min}_\text{RTT}\{42, 08, 15, 64, 32, 07, 64, 16, 08, 08\}
\]
NAS algorithm:

Variables: $\text{min\_rtt} = \text{e.g. } 7\text{ms}, \text{nodes\_copy} = \text{nodes}$

for node in nodes {
    if $\text{min\_rtt} == \text{getRTT(node, pod.location)}$ {
        if $\text{pod.min\_bandwidth} \leq \text{getAvBandwidth(node)}$ {
            return node
        } else {
            nodes\_copy.removeNode(node)
        }  
    }
}
### Working Principle

**NAS algorithm:**

**Minimum Bandwidth:** $\text{Pod}.\text{min\_bandwidth} = 4.0$

<table>
<thead>
<tr>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTT:</td>
<td>42</td>
<td>08</td>
<td>15</td>
<td>64</td>
<td>32</td>
<td>07</td>
<td>64</td>
<td>16</td>
<td>08</td>
</tr>
<tr>
<td>Bandwidth:</td>
<td>5.5</td>
<td>3.7</td>
<td>4.2</td>
<td>4.7</td>
<td>5.3</td>
<td>1.7</td>
<td>1.3</td>
<td>3.8</td>
<td>4.5</td>
</tr>
</tbody>
</table>

The algorithm checks if the bandwidth is less than the minimum bandwidth. If it is, a node is marked as unusable. In this case, Node 07 is marked as unusable because its bandwidth is 1.7, which is less than 4.0.
NAS algorithm:

Variables: \( \text{nodes\_copy} = \text{nodes} \ \setminus \ \{\text{removed\_nodes}\} \)

\[
\begin{align*}
\text{if } \text{nodes\_copy} &= \text{null} \ {\{\text{null}} \text{\}} \\
&\quad \quad \text{return Exception("No suitable nodes found")} \\
\text{else} \ {\} \\
&\quad \quad \text{return NAS(\text{nodes\_copy}, \ pod)} \\
\end{align*}
\]
## NAS algorithm:

**Working Principle**

### NAS algorithm: step 2

<table>
<thead>
<tr>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>08</td>
<td>15</td>
<td>64</td>
<td>32</td>
<td><strong>07</strong></td>
<td>64</td>
<td>16</td>
<td>08</td>
<td>08</td>
</tr>
<tr>
<td>5.5</td>
<td>3.7</td>
<td>4.2</td>
<td>4.7</td>
<td>5.3</td>
<td>1.7</td>
<td>1.3</td>
<td>3.8</td>
<td>4.5</td>
<td>6.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RTT:</th>
<th>Bandwidth:</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>5.5</td>
</tr>
<tr>
<td>08</td>
<td>3.7</td>
</tr>
<tr>
<td>15</td>
<td>4.2</td>
</tr>
<tr>
<td>64</td>
<td>4.7</td>
</tr>
<tr>
<td>32</td>
<td>5.3</td>
</tr>
<tr>
<td><strong>07</strong></td>
<td><strong>1.7</strong></td>
</tr>
<tr>
<td>64</td>
<td>1.3</td>
</tr>
<tr>
<td>16</td>
<td>3.8</td>
</tr>
<tr>
<td>08</td>
<td>4.5</td>
</tr>
<tr>
<td>08</td>
<td>6.8</td>
</tr>
</tbody>
</table>
## NAS algorithm:

### Working Principle

### NAS algorithm: step 3

<table>
<thead>
<tr>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTT:</td>
<td>42</td>
<td>08</td>
<td>15</td>
<td>64</td>
<td>32</td>
<td>64</td>
<td>16</td>
<td>08</td>
</tr>
<tr>
<td>Bandwidth:</td>
<td>5.5</td>
<td>3.7</td>
<td>4.2</td>
<td>4.7</td>
<td>5.3</td>
<td>1.3</td>
<td>3.8</td>
<td>4.5</td>
</tr>
</tbody>
</table>
NAS algorithm:

**Working Principle**

### NAS algorithm: step 1

<table>
<thead>
<tr>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>08</td>
<td>15</td>
<td>64</td>
<td>32</td>
<td>64</td>
<td>16</td>
</tr>
</tbody>
</table>

\[
\text{min}_\text{RTT}\{42, 08, 15, 64, 32, 64, 16, 08, 08\}
\]
**NAS algorithm:**

**Working Principle**

### NAS algorithm: step 2

<table>
<thead>
<tr>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTT: 42</td>
<td>08</td>
<td>15</td>
<td>64</td>
<td>32</td>
<td>64</td>
<td>16</td>
<td>08</td>
</tr>
<tr>
<td>Bandwidth: 5.5</td>
<td>3.7</td>
<td>4.2</td>
<td>4.7</td>
<td>5.3</td>
<td>1.3</td>
<td>3.8</td>
<td>4.5</td>
</tr>
</tbody>
</table>

Minimum Bandwidth: \( \text{Pod}.\min\_\text{bandwidth} = 4.0 \)
### Working Principle

**NAS algorithm:**

<table>
<thead>
<tr>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTT:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>08</td>
<td>15</td>
<td>64</td>
<td>32</td>
<td>64</td>
<td>16</td>
</tr>
<tr>
<td>Bandwidth:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.5</td>
<td>3.7</td>
<td>4.2</td>
<td>4.7</td>
<td>5.3</td>
<td>1.3</td>
<td>3.8</td>
</tr>
</tbody>
</table>

**NAS algorithm: step 2**

- RTT: 08 < 4.0
- Bandwidth: 4.5 > 4.0

**Minimum Bandwidth:** Pod.min_bandwidth = 4.0
NAS algorithm:

<table>
<thead>
<tr>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>08</td>
<td>15</td>
<td>64</td>
<td>32</td>
<td>64</td>
<td>16</td>
<td>08</td>
</tr>
<tr>
<td>5.5</td>
<td>3.7</td>
<td>4.2</td>
<td>4.7</td>
<td>5.3</td>
<td>1.3</td>
<td>3.8</td>
<td>4.5</td>
</tr>
<tr>
<td>6.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RTT: 42 08 15 64 32 64 16 08 08
Bandwidth: 5.5 3.7 4.2 4.7 5.3 1.3 3.8 4.5 6.8
Evaluation:

- compared to Kube-scheduler and Random Scheduler (RS)
  - default of K8s
  - scheduler extender
  - random pick
Results

Evaluation:

- compared to **Kube-scheduler** and **Random Scheduler (RS)**
  - default of K8s
  - scheduler extender
  - random pick
- air quality data
- organic compounds
Results

Evaluation:

Node 1
Node 2
Node 3
Node 4
Node 5
Node 6
Node 7
Node 8
Node 9
Node 10
Node 11
Node 12
Node 13
Node 14

Locations:
Location A
Location B
Location C
Location D
Location E
## Results

### Evaluation:

<table>
<thead>
<tr>
<th>Scheduler</th>
<th>Extender decision</th>
<th>Scheduling decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>KS</td>
<td>-</td>
<td>2.14 ms</td>
</tr>
<tr>
<td>RS</td>
<td>5.32 ms</td>
<td>7.71 ms</td>
</tr>
<tr>
<td>NAS</td>
<td>4.82 ms</td>
<td>6.44 ms</td>
</tr>
</tbody>
</table>

[1]: Average execution time of different schedulers
## Results

### Evaluation:

<table>
<thead>
<tr>
<th>Scheduler</th>
<th>Extender decision</th>
<th>Scheduling decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>KS</td>
<td>-</td>
<td>2.14 ms</td>
</tr>
<tr>
<td>RS</td>
<td>5.32 ms</td>
<td>7.71 ms</td>
</tr>
<tr>
<td>NAS</td>
<td>4.82 ms</td>
<td>6.44 ms</td>
</tr>
</tbody>
</table>

[1]: Average execution time of different schedulers
## Results

### Evaluation:

<table>
<thead>
<tr>
<th>Locations</th>
<th>RS</th>
<th>KS</th>
<th>NAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Number of Pods**

<table>
<thead>
<tr>
<th>Locations</th>
<th>Service: 1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>RS</td>
<td>KS</td>
<td>NAS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Page: 63/74
## Results

### Evaluation:

<table>
<thead>
<tr>
<th>Locations</th>
<th>RS</th>
<th>KS</th>
<th>NAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Locations</th>
<th>number of Pods</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
</tr>
</tbody>
</table>

### Service:

- 1: A
- 2: A
- 3: B
- 4: C
- 5: C
- 6: D

---

Page: 64/74
## Results

### Evaluation:

<table>
<thead>
<tr>
<th>Locations</th>
<th>RS</th>
<th>KS</th>
<th>NAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Number of Pods:**

- **A:**
  - RS: 0
  - KS: 1
  - NAS: 3
- **B:**
  - RS: 1
  - KS: 2
  - NAS: 1
- **C:**
  - RS: 1
  - KS: 2
  - NAS: 3
- **D:**
  - RS: 1
  - KS: 0
  - NAS: 3
- **E:**
  - RS: 0
  - KS: 0
  - NAS: 0

### Service:

- **1** (Green): A
- **2** (Orange): A
- **3** (Purple): B
- **4** (Dark Gray): C
- **5** (Green): C
- **6** (Green): D

Page: 65/74
Results

Evaluation:

Average RTT of different schedulers

≈ 80% smaller

[1]: Average RTT of different schedulers
## Results

### Comparison:

<table>
<thead>
<tr>
<th>Kube-scheduler (KS)</th>
<th>Network-Aware Scheduler (NAS)</th>
<th>Random Scheduler (RS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>- no extender call</td>
<td>- aware of network:</td>
<td>- unaware of network</td>
</tr>
<tr>
<td></td>
<td>• network latency</td>
<td>• extender call</td>
</tr>
<tr>
<td></td>
<td>• bandwidth</td>
<td>• RTT ▲</td>
</tr>
<tr>
<td></td>
<td>• 80% RTT ▼</td>
<td>• desired location ▲</td>
</tr>
<tr>
<td></td>
<td>• desired location ▲</td>
<td>• desired location ▼</td>
</tr>
<tr>
<td></td>
<td>• doesn’t overload nodes</td>
<td>• overloads nodes</td>
</tr>
<tr>
<td></td>
<td>→ Smart Cities ▼</td>
<td>→ Smart Cities ▼</td>
</tr>
<tr>
<td>- unaware of network</td>
<td>- extender call</td>
<td>-</td>
</tr>
<tr>
<td>- RTT ▲</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>- desired location ▼</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>- overloads nodes</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>→ Smart Cities ▼</td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

... but can still be improved
Thank you for your attention! 😊
Images:
[fog]: By Brocken Inaglory - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=7908286
[clo]: By Jessie Eastland - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=22865059
[sma]: By Massimo Catarinella - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4553808
Sources

Images:
[com]: By Everaldo Coelho (YellowIcon); - All Crystal icons were posted by the author as LGPL on kde-look, the SVG version in various KDE packages;, LGPL, https://commons.wikimedia.org/w/index.php?curid=69374935
[ant]: By Fuss - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1449342
Sources

Images:
Knowledge:
Sources

Knowledge:


Knowledge:
[9]: https://vitux.com/how-to-check-if-your-processor-supports-virtualization-technology/