EcoFaaS: Rethinking the Design of Serverless Environments for Energy Efficiency

ISCA 2024

Jovan Stojkovic, Nikoleta Iliakopoulou, Tianyin Xu, Hubertus Franke*, Josep Torrellas
University of Illinois at Urbana-Champaign
*IBM Research
What is serverless computing?

- Serverless computing popular cloud paradigm
  - Users deploy applications, providers provision resources
- Many benefits
  - Simple and modular programming
  - Automatic resource scaling
  - Pay-as-you-go model
- AWS Lambda, Microsoft Azure, IBM Cloud
How serverless computing works?

Invoke function

Deploy function

Node 1
Container 1
Runtime Func A

Container 2
Runtime Func A

Container 3
Runtime Func B

Node Controller

Node 2
Container 4
Runtime Func B

Container 5
Runtime Func B

Container 6
Runtime Func C

Node Controller

Frontend
Load Balancer
Functions

Node 1

Node 2
How serverless computing works?

Energy-efficiency?
Contributions

- Characterization of energy-efficiency in serverless environments
- Propose **EcoFaaS**
  - Service Level Objective (SLO) driven energy-efficient serverless system
- Reduces energy consumption by 42%, and tail-latency by 35%
How to Save Energy?

- Obvious approach: voltage-frequency scaling (DVFS)?
How to Save Energy?

- Obvious approach: voltage-frequency scaling (DVFS)?

What are the challenges towards energy-efficient serverless systems?
Serverless Functions are Highly Diverse
Serverless Functions are Highly Diverse
Serverless Functions are Highly Diverse

In energy-efficient serverless environments, users need to only specify the end-to-end SLOs.
Idle Time Dominates Function Execution

Request
- Prepare request
- Download image
- Resize image
- Upload image
- Prepare response

Response

Function Storage

Busy
Waiting
Idle Time Dominates Function Execution

Example diagram showing the execution flow of different functions.
Idle Time Dominates Function Execution
Idle Time Dominates Function Execution
Idle Time Dominates Function Execution
Idle Time Dominates Function Execution
Idle Time Dominates Function Execution

Request Queue
Idle Time Dominates Function Execution

Energy-efficient serverless environments need to exploit the abundant idle time.
Frequency Changes are Expensive

Function invocation

Idle → RPC

Frequent context switches!
Frequency Changes are Expensive

Function invocation

Frequent changes of core frequency!
Frequency Changes are Expensive

Energy-efficient serverless environments need to minimize frequency changes.
Serverless Workloads are Highly Dynamic
Serverless Workloads are Highly Dynamic
Serverless Workloads are Highly Dynamic
Serverless Workloads are Highly Dynamic

Energy-efficient serverless environments need to dynamically adjust to the fluctuating workloads.
Conventional Systems are not Optimized for Energy-Efficiency
Conventional Systems are not Optimized for Energy-Efficiency
Conventional Systems are not Optimized for Energy-Efficiency
Conventional Systems are not Optimized for Energy-Efficiency
EcoFaaS: An Energy Management Framework for Serverless Environments
EcoFaaS Key Ideas

1. EcoFaaS is driven by SLO metrics
EcoFaaS: An Energy Management Framework for Serverless Environments
Choosing CPU Frequency with EcoFaaS

Delay-Power Table

<table>
<thead>
<tr>
<th></th>
<th>Fa</th>
<th>Fb</th>
<th>Fc</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_1</td>
<td>t_{1a}, E_{1a}</td>
<td>t_{1b}, E_{1b}</td>
<td>t_{1c}, E_{1c}</td>
</tr>
<tr>
<td>f_2</td>
<td>t_{2a}, E_{2a}</td>
<td>t_{2b}, E_{2b}</td>
<td>t_{2c}, E_{2c}</td>
</tr>
<tr>
<td>f_3</td>
<td>t_{3a}, E_{3a}</td>
<td>t_{3b}, E_{3b}</td>
<td>t_{3c}, E_{3c}</td>
</tr>
</tbody>
</table>
Choosing CPU Frequency with EcoFaaS

MILP in Workflow Controller

\[
\begin{align*}
\min & \quad \sum E_i \\
\text{s.t.} & \quad \sum (ti) \leq SLO
\end{align*}
\]
Choosing CPU Frequency with EcoFaaS
1. EcoFaaS is driven by SLO metrics
2. EcoFaaS profiles and predicts the execution time and energy of function invocations
EcoFaaS: An Energy Management Framework for Serverless Environments
Scheduling Requests with EcoFaaS

Container $FuncB$

Request + SLO
Scheduling Requests with EcoFaaS

<table>
<thead>
<tr>
<th>Container</th>
<th>FuncB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispatcher</td>
<td>Handlers</td>
</tr>
</tbody>
</table>

- Dispatchers: Normal
- Handlers: Normal, Red

39
### Scheduling Requests with EcoFaaS

<table>
<thead>
<tr>
<th>Container</th>
<th>FuncB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispatcher</td>
<td>Handlers</td>
</tr>
</tbody>
</table>

Execution time?  
Idle time?  
Energy?
Scheduling Requests with EcoFaaS

Container *FuncB*

| Dispatcher | Handlers |

Core Pool

Queuing time?

Core Pool

Queuing time?
Scheduling Requests with EcoFaaS
EcoFaaS Key Ideas

1. EcoFaaS is driven by SLO metrics
2. EcoFaaS profiles and predicts the execution time and energy of function invocations
3. EcoFaaS splits cores into frequency classes
EcoFaaS: An Energy Management Framework for Serverless Environments

Invoke function

Node 1
- Node Controller
  - FuncA
  - FuncX
  - FuncB
  - Pool Scheduler

Node 2
- Node Controller
  - FuncY
  - FuncZ
  - FuncC
  - Pool Scheduler

Workflow Controller

Load Balancer

Storage
Scheduling Requests with EcoFaaS

Frequency Pool Scheduler

Core Pool

Core Pool
Scheduling Requests with EcoFaaS

Core Pool

Frequency Pool Scheduler

Core Pool

46
Scheduling Requests with EcoFaaS

Core Pool

Frequency Pool Scheduler

Core Pool
Scheduling Requests with EcoFaaS

Core Pool

Frequency Pool Scheduler

Core Pool
Scheduling Requests with EcoFaaS

Frequency Pool Scheduler

Core Pool

No frequency changes!
EcoFaaS Key Ideas

1. EcoFaaS is driven by SLO metrics
2. EcoFaaS profiles and predicts the execution time and energy of function invocations
3. EcoFaaS splits cores into frequency classes
4. EcoFaaS changes pools and pool frequencies dynamically
EcoFaaS: An Energy Management Framework for Serverless Environments
Re-configuring the Pools with EcoFaaS

Core Pool
Frequency Pool Scheduler

Node Controller

Core Pool
Frequency Pool Scheduler
Re-configuring the Pools with EcoFaaS
Re-configuring the Pools with EcoFaaS

Node Controller

# cores, frequency

Core Pool

Frequency Pool Scheduler

Core Pool

Frequency Pool Scheduler

# cores, frequency
Evaluation Methodology

- Cluster with 5 Intel Xeon servers (20 cores, each)
- Platforms: OpenWhisk and KNative
- Systems evaluated
  - **Baseline**: state-of-the-art serverless platform (MXFaaS ISCA’23)
  - **Baseline+PowerCtrl**: Baseline + state-of-the-art power management (Gemini MICRO’20)
  - **EcoFaaS**: our proposal
EcoFaaS Reduces Energy Consumption

![Energy Consumption Chart]

- **Baseline**
- **Baseline+PowerCtrl**
- **EcoFaaS**

The chart shows a significant reduction in energy consumption with EcoFaaS compared to the baseline and baseline with power control.
EcoFaaS Reduces Energy Consumption

Energy reduction by 42% over state-of-the-art baseline!
EcoFaaS Reduces Tail Latency

![Graph showing the comparison of Baseline, Baseline+PowerCtrl, and EcoFaaS for Norm Tail Latency. The graph indicates that EcoFaaS significantly reduces tail latency compared to the baseline and Baseline+PowerCtrl.]
EcoFaaS Reduces Tail Latency

Tail latency reduction by 35% over state-of-the-art baseline!
Conclusion

- Serverless computing beneficial, but current execution energy-inefficient
- Propose **EcoFaaS** – SLO-driven energy-efficient serverless system
- Reduces energy consumption by 42%, and tail-latency by 35%
EcoFaaS: Rethinking the Design of Serverless Environments for Energy Efficiency

ISCA 2024

Jovan Stojkovic, Nikoleta Iliakopoulou, Tianyin Xu, Hubertus Franke*, Josep Torrellas
University of Illinois at Urbana-Champaign
*IBM Research