SUPPLEMENTARY MATERIAL

Title – SHIELD: A Secure Heuristic Integrated Environment for Load Distribution in Rural-AI

Journal: Elsevier Future Generation Computer Systems

Authors: Ashish Kaushal, Osama Almurshed, Osama Almoghamis, Areej Alabbas, Nitin Auluck, Bharadwaj Veeravalli, and Omer Rana.

1 Analysing Performance with Different Data Distribution

The primary experimentation in the work has been performed with uniform data distribution but in order to evaluate the framework for a different probability of task arrival, we analysed the performance of SHIELD over Gaussian distribution of data.

Figures from 1 to 6 below shows the failure rate, makespan, and utilisation of all the three workflows on Parsl and OpenWhisk platform with Gaussian distribution of data. We observed that on global workflow, the failure rate increased to 6.20% and 9.40% for Parsl and OpenWhisk respectively. Similarly, for local and prediction workflows, the failure rates also increased to 26.27%, 23.93%; 0.07%, 2.47% respectively on Parsl and OpenWhisk. We also noticed that the makespan for both the distributions were similar with slight variation of average 1s-3s. Gaussian distribution simulates a more realistic and varied workload scenario where some nodes might be heavily loaded than others; SHIELD still outperformed the other strategies. However, the slight increase in failure rate can be observed due to changing load pattern of the distribution which can be more challenging than managing a consistent, evenly spread-out load.

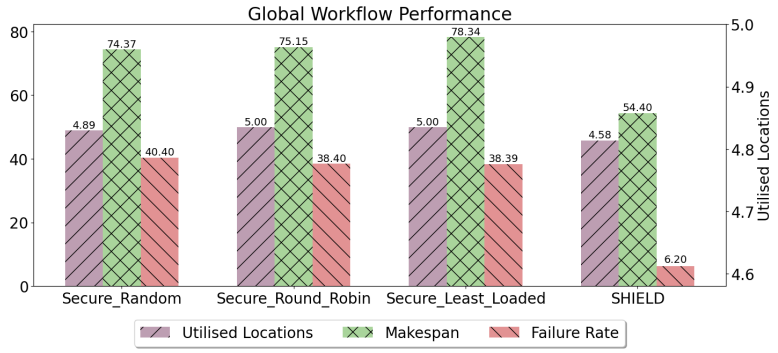

Figure 1: Global workflow evaluation on Parsl platform (with Gaussian Distribution)

2 Analysing Performance with Variable Task Load

The primary experimentation in this work has been performed with total 5000 tasks (out of which 1500 were local requests, 500 were global requests, and 3500 were prediction requests). In order to test the SHIELD framework with different load, we have increased the task load to 5x and 20x. The experimentation shows a notable trend in both Parsl and OpenWhisk platforms. In most cases, the makespan marginally increases or remains stable, indicating a robust handling of larger workloads by SHIELD framework. The utilised locations slightly vary, reflecting adaptive resource allocation strategies to accommodate the increased load. Interestingly, the failure rate increases (around 1-10%, depending on type of workflow) with higher task load, suggesting a correlation between increased workload and the likelihood of task failures.

The results obtained from the experiment is shown below in Table 1:

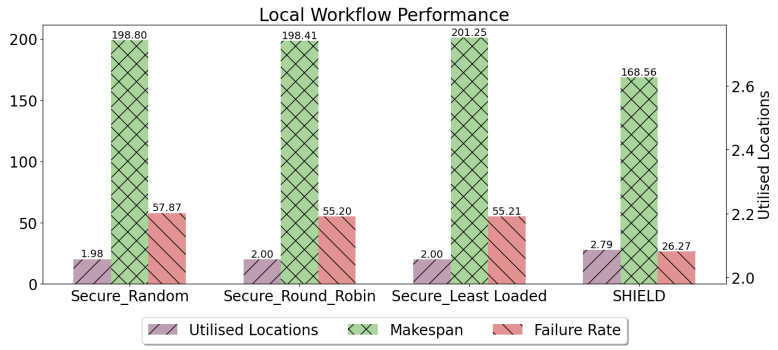

Figure 2: Local workflow evaluated on Parsl platform (with Gaussian Distribution)

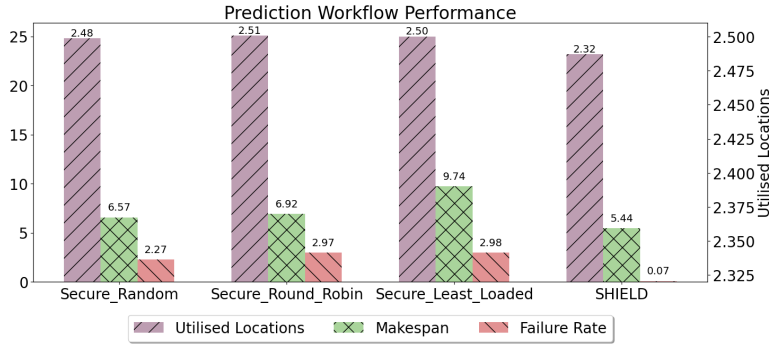

Figure 3: Prediction workflow evaluated on Parsl platform (with Gaussian Distribution)

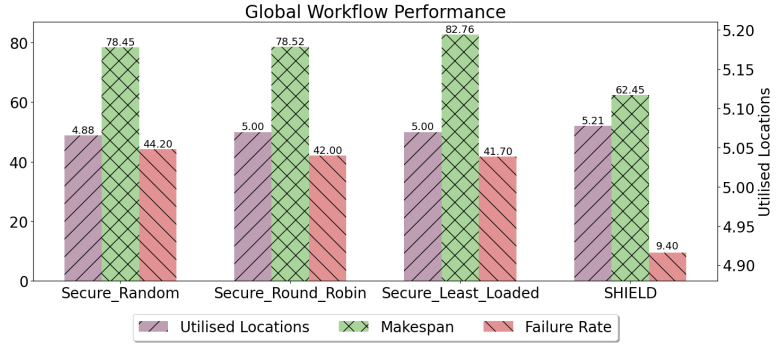

Figure 4: Global workflow evaluation on OpenWhisk platform (with Gaussian Distribution)

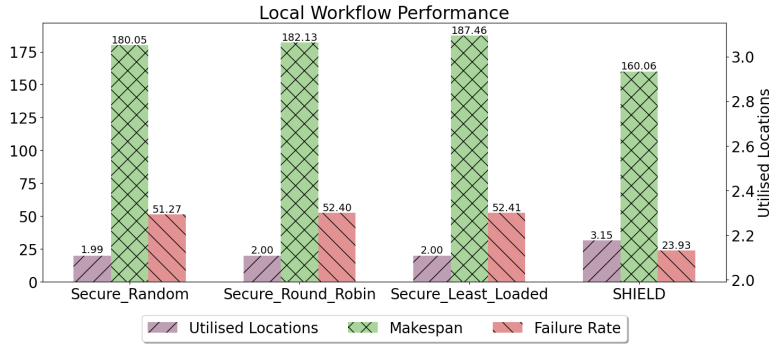

Figure 5: Local workflow evaluated on OpenWhisk platform (with Gaussian Distribution)

3 Analysing Performance with Heterogeneous Resource Environment

In order to analyse performance in a heterogeneous resource environment, we have evaluated SHIELD framework with two types of hardware resources. We considered a Raspberry Pi 4, Quad core Cortex-A72 Processor, 64-bit SoC @1.8GHz

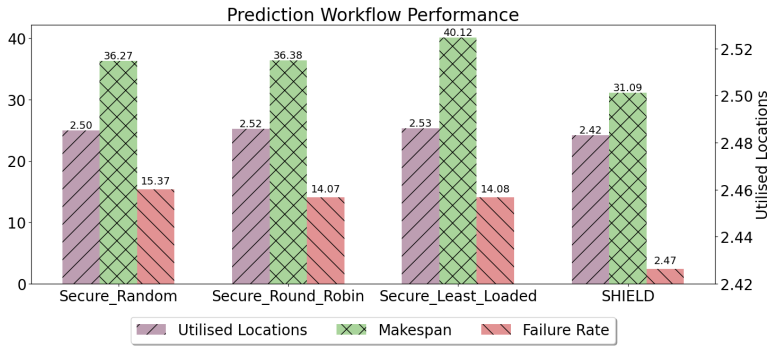

Figure 6: Prediction workflow evaluated on OpenWhisk platform (with Gaussian Distribution)

Platform	Workflow	Requests Load	Makespan	Failure Rate	Utilised Locations
Parsl	local	5x	169.16	21.52	2.58
Parsl	local	20x	170.39	26.16	2.45
Parsl	global	5x	54.24	6.68	4.91
Parsl	global	20x	54.90	6.87	4.65
Parsl	prediction	5x	5.44	0.01	2.26
Parsl	prediction	20x	5.44	0.07	2.32
OpenWhisk	local	5x	160.11	22.61	2.92
OpenWhisk	local	20x	159.57	23.05	2.84
OpenWhisk	global	5x	63.71	10.88	4.68
OpenWhisk	global	20x	63.95	9.16	4.71
OpenWhisk	prediction	5x	30.98	2.22	2.34
OpenWhisk	prediction	20x	31.02	2.23	2.31

Table 1: Comparative analysis of performance metrics across various platforms: Evaluating SHIELD framework with 5x and 20x task load

having 4GB RAM and NVIDIA Jetson Nano, Quad-core ARM Cortex-A57 MPCore Processor with 4GB RAM. We observed that SHIELD outperforms other approaches in utilising locations (4.42, 4.63) more effectively, achieving lower makespan (81.98s, 85.48s), and ensuring a lower failure rate (9.73%, 8.75%) across both Parsl and OpenWhisk platforms respectively. In this setting, the experimentation demonstrate that Parsl is a better choice for global and prediction workflows as it shows faster task completion (6.37s, 26.27s) and high reliability (2%, 1.6%). Its architecture and execution model are well-suited for complex computations and scenarios where faster execution and success rates are critical for performance. However, OpenWhisk shows better performance than Parsl in local workflow where pre-allocating the resources before task execution can significantly improves the makespan (22.14s) and reduces the failure rate (4.28%).

Comparing the performance of SHIELD framework when evaluated on a uniform mix of resources, we observed that similar trends across all workflows and both the platforms is achieved. The results show that the resource utilisation in both the scenarios is almost similar whereas there is a slight increase in makespan and failure rate when execution is performed in heterogeneous environment. This increase can be attributed to the overheads incurred from serverless architecture and GPU utilisation in the edge layer. In the serverless deployments, as seen with both Parsl and OpenWhisk, there are significant initialisation overheads, more dominantly seen in GPUs due to their complexity and higher memory requirements. Moreover, GPUs do not have the advantage of spreading out initialisation costs over longer duration of periods since functions are temporary and do not run continuously. This results in notable delays every time the GPU is reloaded, thus resulting in higher execution cost during task execution. Furthermore, GPU performance is also affected by memory hierarchy challenges, as frequent data transfers between different memory layers lead to bottlenecks, particularly in devices with limited memory in GPUs such as the Jetson Nano's used in edge computing infrastructures.

The following six figures (from Figure 7 to Figure 12) below shows the experimentation results on three different workflows (global, local, prediction) evaluated on two different platforms (parsl and openwhisk).

4 Critical Observations in the Results

In our analysis of the SHIELD framework, we have observed several key insights. Firstly, SHIELD demonstrates a notable increase in resource utilization efficiency, evident through reduced makespan and lower failure rates compared to

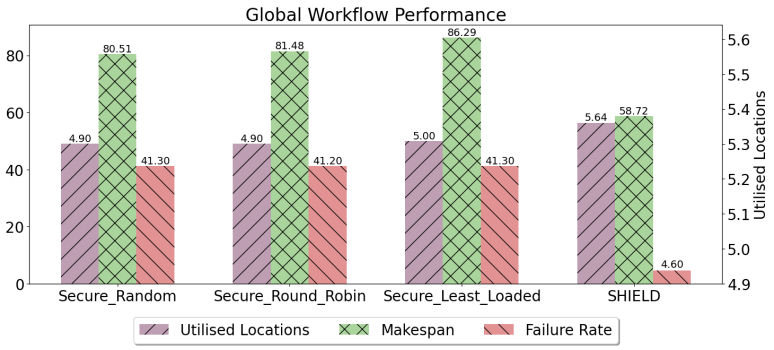

Figure 7: Global workflow evaluation on Parsl platform

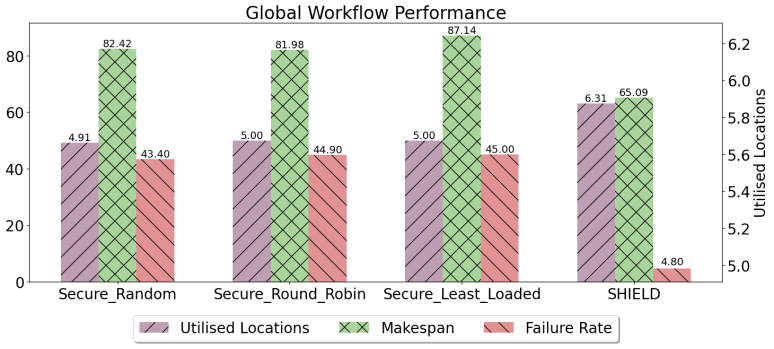

Figure 8: Global workflow evaluation on OpenWhisk

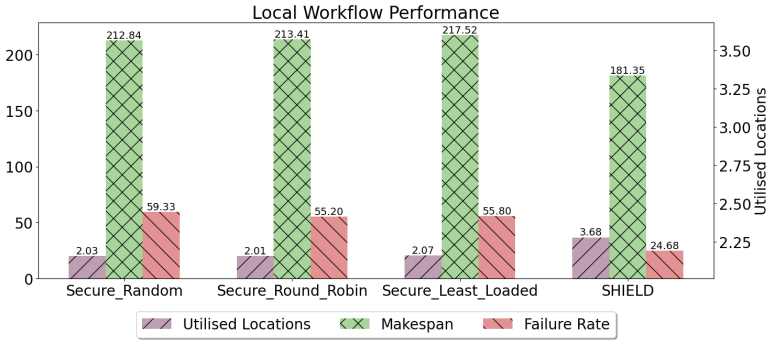

Figure 9: Local workflow evaluation on Parsl platform

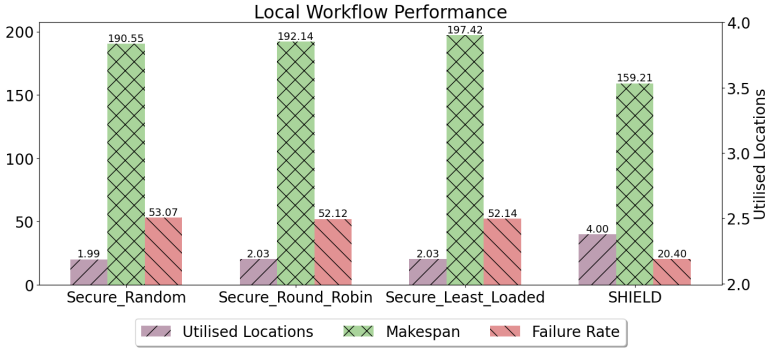

Figure 10: Local workflow evaluation on OpenWhisk

other methods, indicating its superior performance in environments with limited resources. This is particularly crucial in applications like precision agriculture, where timely and accurate data processing is highly important. The framework's

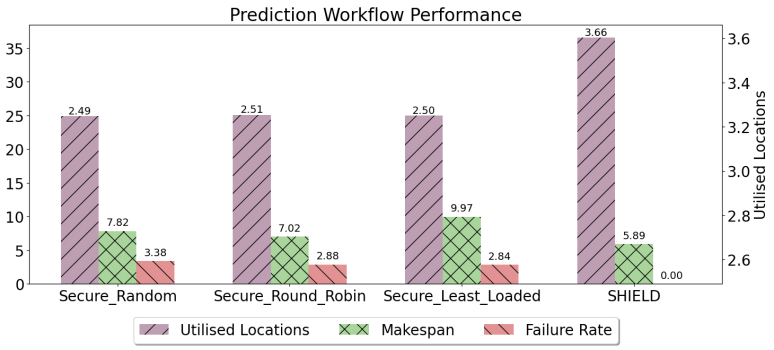

Figure 11: Prediction workflow evaluation on Parsl platform

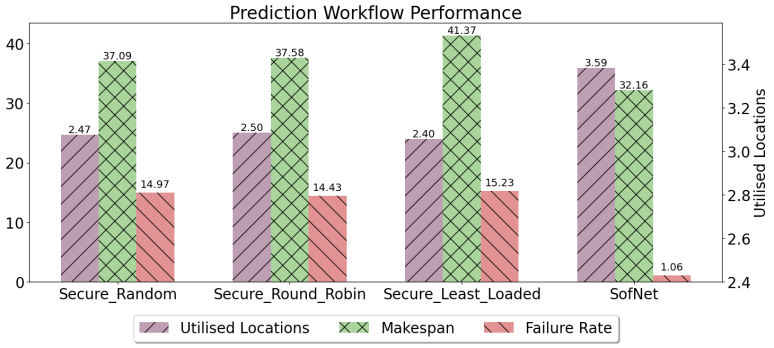

Figure 12: Prediction workflow evaluation on OpenWhisk

adaptability is highlighted by its dynamic resource allocation capabilities, adjusting effectively to varying operational demands as shown in different workflow scenarios (global, local, and prediction). Additionally, SHIELD strategically prioritises performance and reliability over uniform task distribution, a critical feature in settings with fluctuating resource availability and network stability. These attributes not only illustrate the robustness of SHIELD in managing computational tasks in resource-limited environments but also suggest its potential applicability in a wider range of sectors beyond agriculture, such as smart healthcare and industries.