Online Contextual System Tuning with Bayesian Optimization and Workload Forecasting

Candidato: Luca Moroldo

Supervisor: Prof. Nicola Ferro
Co-Supervisor: Stefano Cereda

21 Feb. 2022

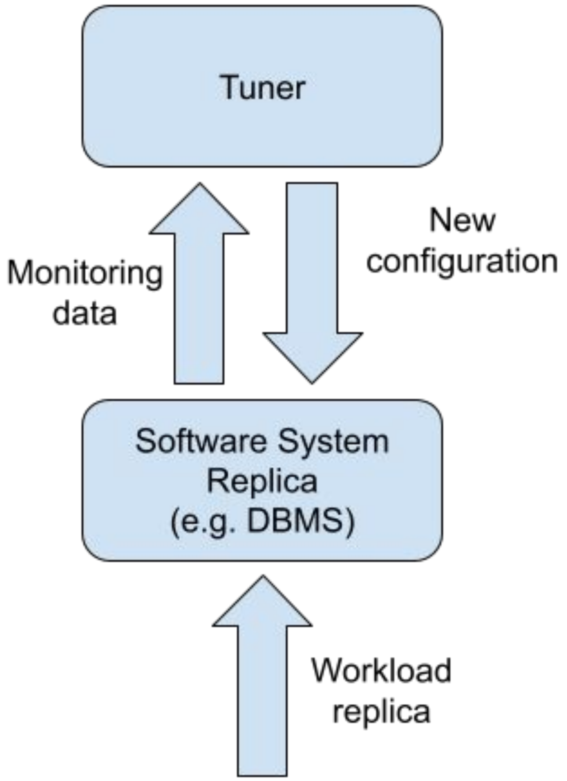
Context

- Akamas **optimizes software systems**, suggesting configurations **x** that minimize/maximize an objective function:

$$\boldsymbol{x}^\star = \operatorname*{argmax}_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$$

- The optimal configuration depends on the system workload

- Applications:
  - Minimize system latency
  - Maximize throughput
  - Minimize infrastructure costs maintaining the same quality of services
- Akamas is based on **Bayesian Optimization** with Gaussian Processes.

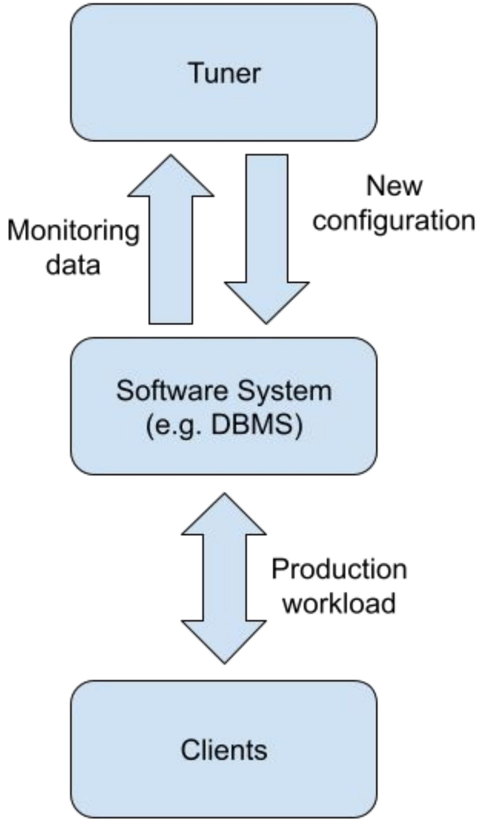


Goal

- **Apply Akamas directly to the production system**, while it is being used by its clients.
- Removing the necessity of replicating both the software system and the workload.

Consequence: Akamas becomes easier to apply and more flexible.

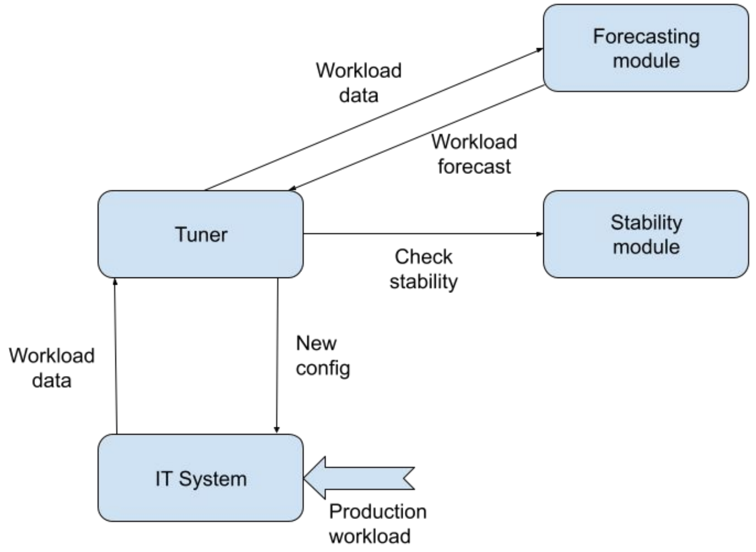
Risks: testing new configurations on a system while it is being used can lead to disservices.



Approccio

The solution is made of 3 modules that integrates with the tuner:

- A **workload forecasting** module
- A **stability** module
- A **workload characterization** module



Workload forecasting and stability

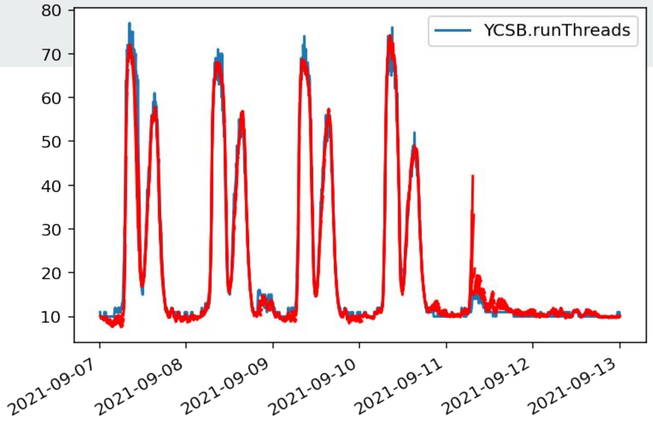
The workload is described by one or more time series (e.g. requests per minute)

Two forecasting models have been chosen and adapted to make forecasts with an horizon of <1 hour:

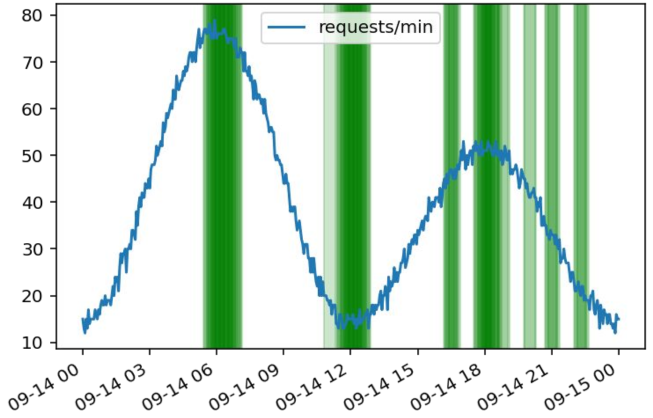
- **Prophet**: based on traditional techniques (GAM and Fuorier series).
- **DeepAR**: uses neural networks, it is based on Sequence to Sequence (S2S) with LSTM cells.

The stability check is done by deterministic algorithms parametrized by a threshold that implement the following stability function:

$$s_\Theta(\tilde{Y}_{t_1:t_2}, Y_{t_0:t_1}) = \wedge_{i=1}^{n} s_\Theta(\tilde{\boldsymbol{y}}^i_{t_1:t_2}, \boldsymbol{y}^i_{t_0:t_1})$$



Requests/min forecasting with DeepAR (red)



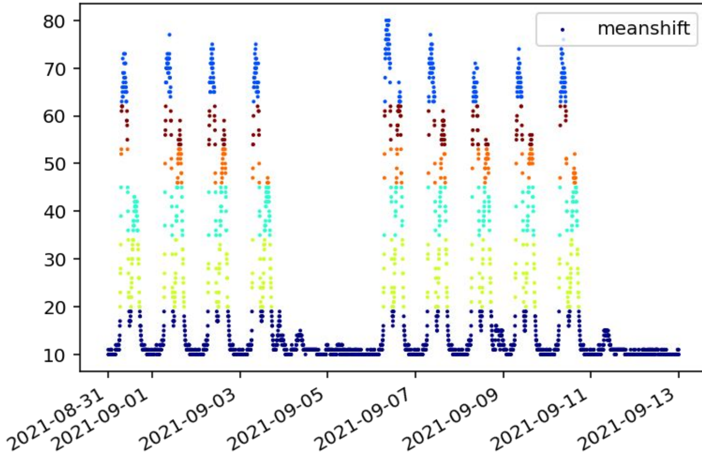
Stable windows (green)

Workload characterization

The workload is automatically characterized with **clustering techniques**.

Two techniques

- **K-means** with Silhouette score to automatically determine the number of clusters.
- **Mean shift** with k-nearest neighbors to automatically determine the bandwidth.



Mean shift clustering of the requests per minute (one color per cluster)

Esperimenti

The proposed solution aims at:

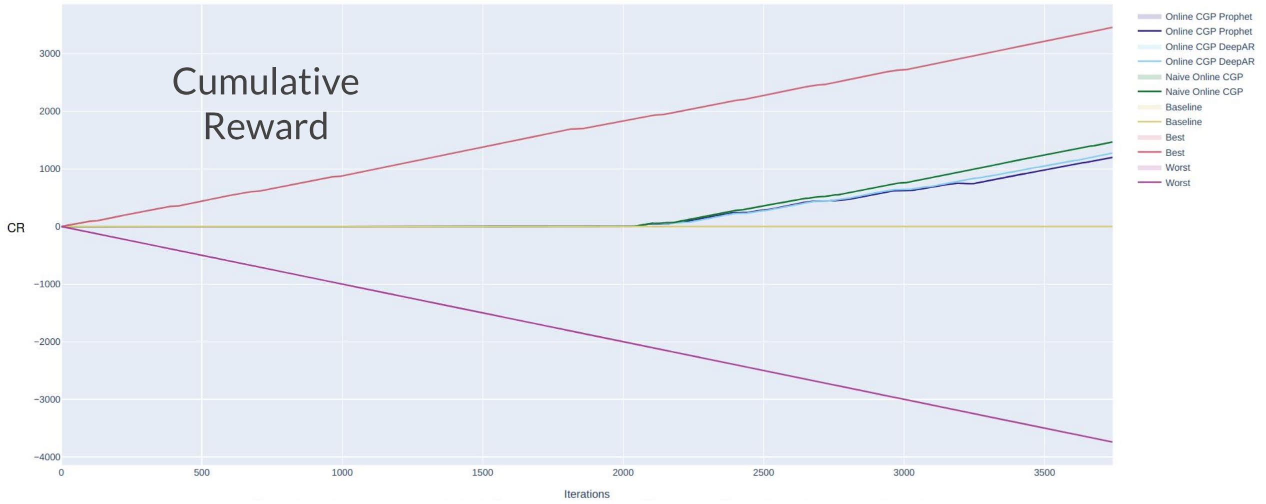
- **Maximize the Cumulative Reward** (CR), that measures the ability of the tuner to suggest workload-relevant configurations.

- **Minimize the number of Failures** (F), caused by inabilities of the system to respond to the incoming requests or constraint violations (e.g. response time above 15ms).

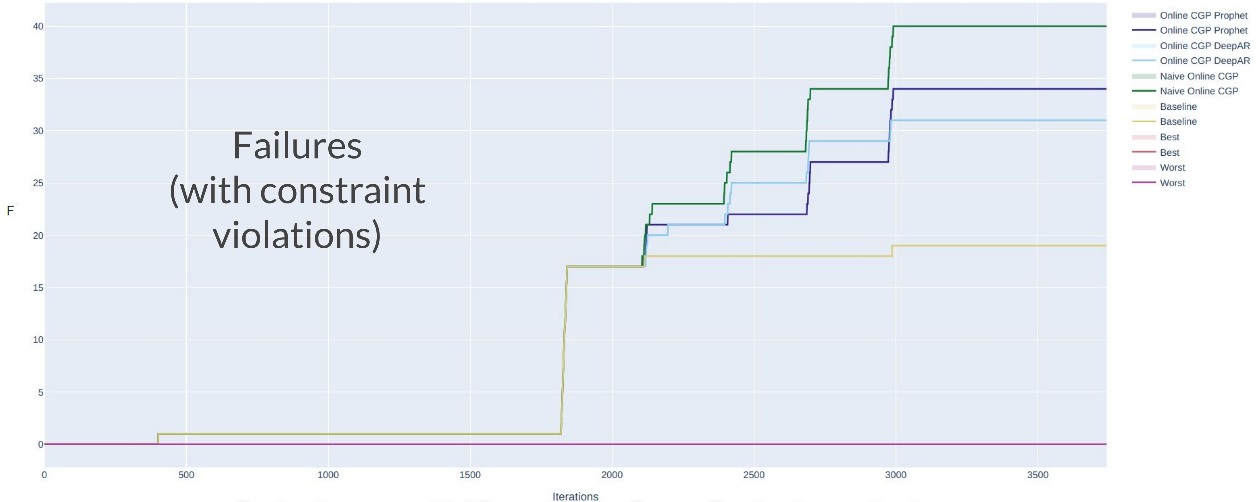
The solution has been tested with 20 scenarios that tune two DBMS models (MongoDB and Cassandra).

Scenario: minimize MongoDB memory usage keeping response time below 10ms.

Note: tuning starts at iteration ~2000.



Yellow: default configuration
Green: tuner without forecasting
Lightblue: tuner with forecasting (DeepAR)
Blue: tuner with forecasting(Prophet)



DBMS	Obj. function	Constraint	Tuner	CR	F
MongoDB	min(memory)	< 9ms latency	Default cfg.	0	3.5%
			No forecasting	71%	3%
			Prophet	61%	2.5%
			DeepAR	**76%**	**2.4%**
MongoDB	min(memory)	< 10ms latency	Default cfg.	0	6.0%
			No foreasting	**47%**	2.7%
			Prophet	32%	2.9%
			DeepAR	45%	**1.7%**
Cassandra	min(latency)	< 1.9 GB memory	Default cfg.	0	0
			No forecasting	**24.4%**	1.0%
			Prophet	21%	**0.7%**
			DeepAR	24%	0.8%
Cassandra	min(latency)	< 1.9 GB memory	Default cfg.	0	0
			No forecasting	43%	0.6%
			Prophet	40%	**<0.1%**
			DeepAR	**45%**	0.1%

Conclusions and future work

- The developed **tuners have always been able to find good configurations** that improve the performance of the default configuration.
- **The usage of forecasting models made the tuning process safer** both in terms of failures and system constraints.

Future work:

- To use the forecasting models to **proactively apply the best configuration found** for the predicted workload.