Online Contextual System Tuning with Bayesian Optimization and Workload Forecasting

Candidato: Luca Moroldo

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

21 Feb. 2022

Contesto

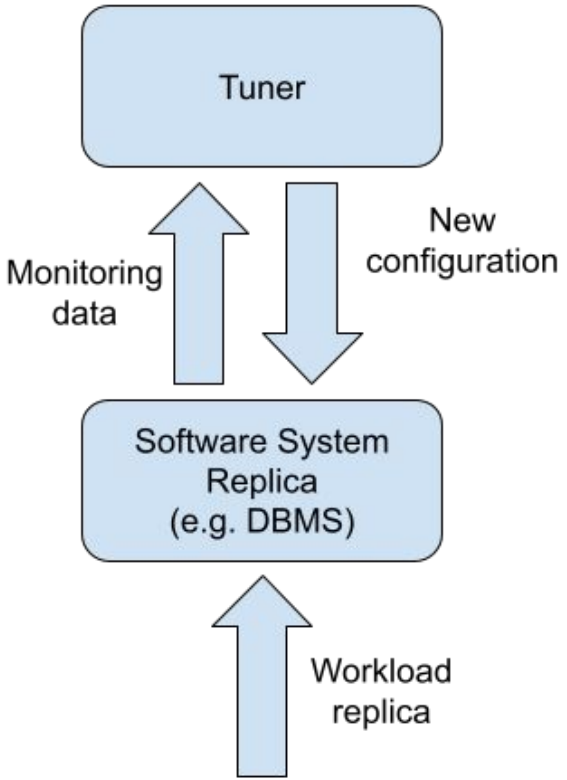
- Akamas **ottimizza sistemi informatici**, suggerendo configurazioni **x** che minimizzano/massimizzano una funzione obiettivo:

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

- La configurazione ottimale dipende dal carico di lavoro

- Esempi applicativi:
  - minimizzare il tempo di risposta
  - massimizzare il throughput
  - minimizzare i costi a parità di qualità dei servizi

- Akamas si basa su **Ottimizzazione Bayesiana** con Processi Gaussiani

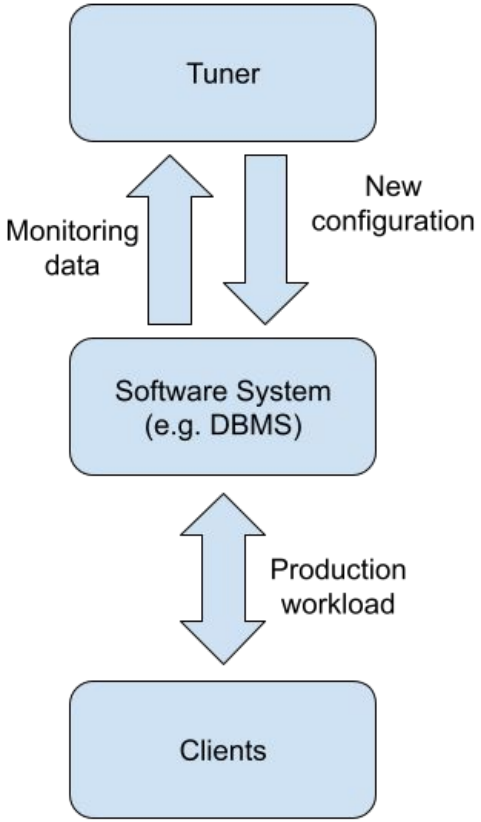


Obiettivo

- Portare Akamas ad **ottimizzare direttamente il sistema di produzione** mentre viene utilizzato dai clienti
- Rimuovendo la necessità di replicare un contesto verosimile del sistema (replica e carico di lavoro)

Conseguenza: minore sforzo nell'applicare Akamas, che diventa più flessibile.

Rischi: sperimentare nuove configurazioni sul sistema mentre viene utilizzato può rovinare l'esperienza utente.

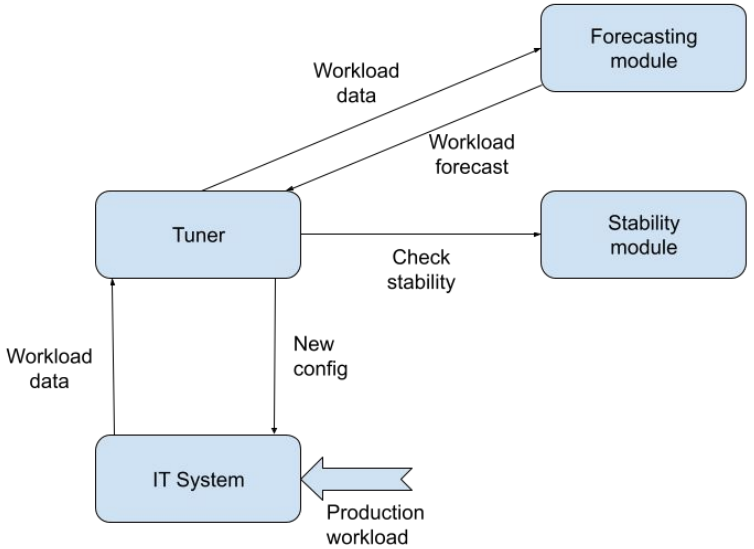


Approccio

La soluzione ha previsto lo sviluppo di 3 moduli che si integrano con l'ottimizzatore:

- Un modulo di **predizione del carico di lavoro**
- Un modulo di **verifica della stabilità**

- Un modulo di **caratterizzazione del carico di lavoro**



Predizione del carico e stabilità

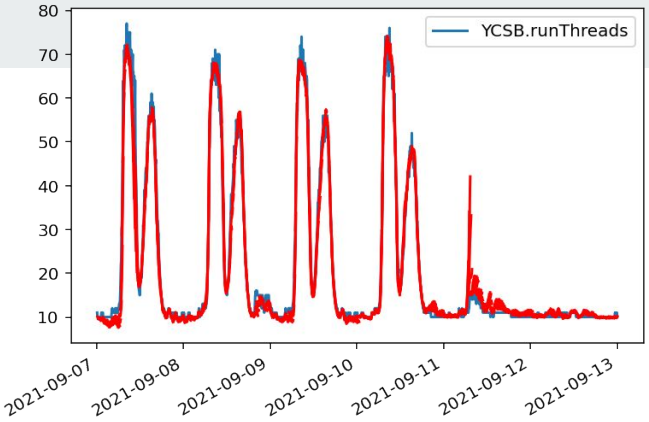
Il carico di lavoro è descritto da una o più serie temporali (e.g. richieste al minuto).

Sono stati utilizzati due modelli di predizione, perfezionati per fare predizioni con un orizzonte di < 1 ora:

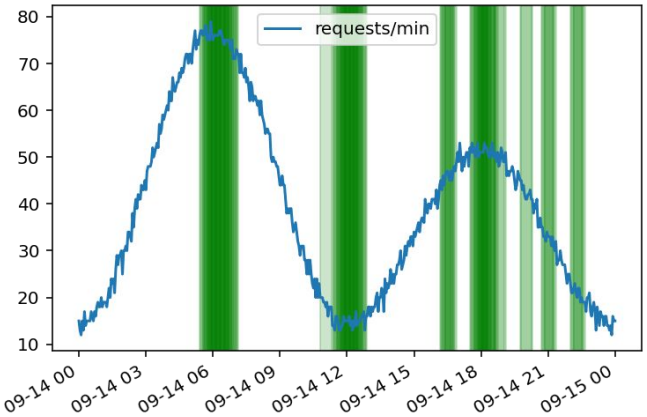
- **Prophet**: utilizza tecniche tradizionali (GAM e serie di Fuorier).
- **DeepAR**: utilizza reti neurali, è basato su Sequence to Sequence (S2S) con celle LSTM.

La verifica di stabilità è svolta con algoritmi deterministici parametrizzati da una soglia, i quali implementano la funzione di "stabilità":

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



Predizione richieste/min DeepAR (rosso)



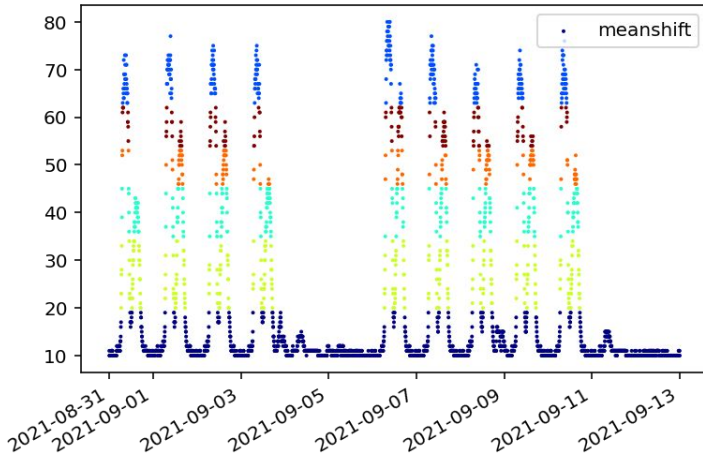
Finestre stabili (verde)

Caratterizzazione del carico di lavoro

Il carico di lavoro è caratterizzato automaticamente tramite **tecniche di clustering**.

Due metodologie:

- **K-means** con punteggio Silhouette per determinare automaticamente il numero di cluster.
- **Mean shift** con k-nearest neighbors per determinare automaticamente la larghezza di banda.



Mean shift clustering delle richieste al minuto (un colore per cluster)

Esperimenti

La soluzione mira a:

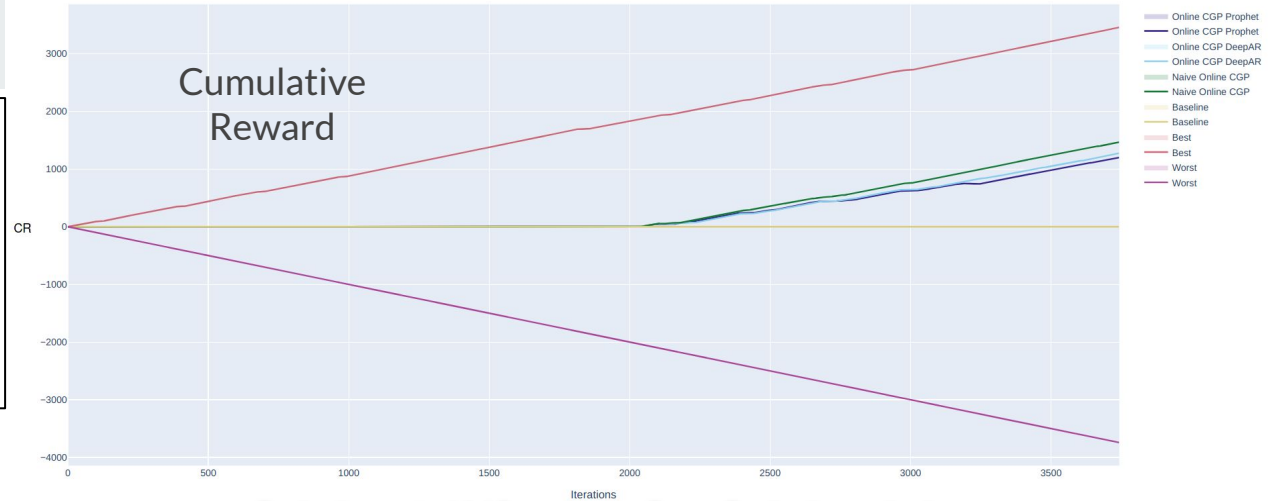
- **Massimizzare il Cumulative Reward** (CR), che misura la capacità dell'ottimizzatore nel suggerire configurazioni in accordo con il carico di lavoro.

- **Minimizzare il numero di Failures** (F), causate da inabilità del sistema nel servire le richieste o da violazioni di vincoli (e.g. tempo di risposta superiore a 15ms).

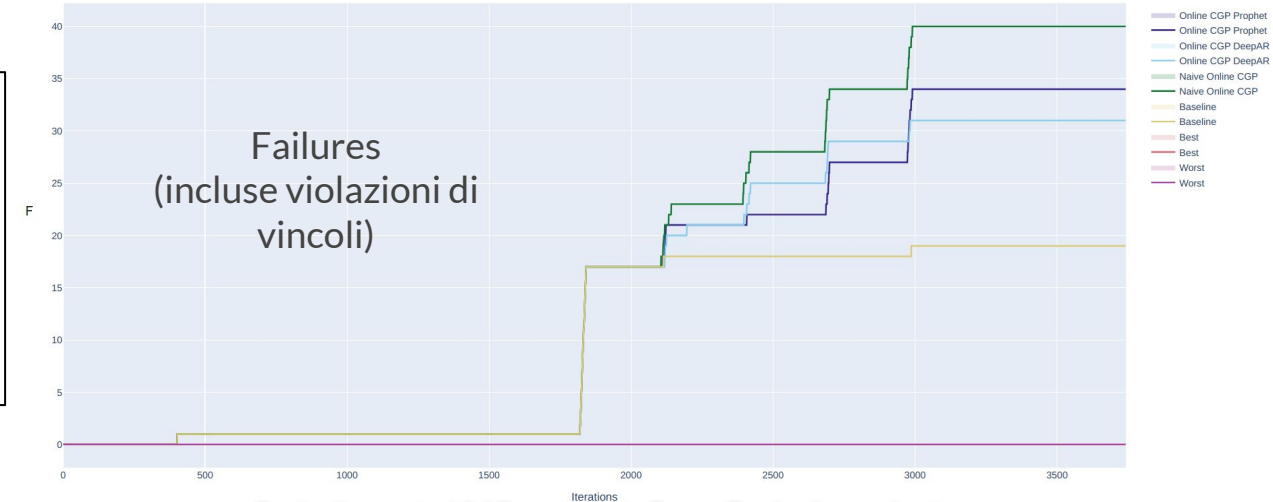
La soluzione è stata verificata con 20 scenari che ottimizzano 2 modelli di DBMS (MongoDB e Cassandra).

Scenario: minimizzare l'utilizzo della memoria di MongoDB mantenendo il tempo di risposta < 10ms.

Nota: l'ottimizzazione inizia all'iterazione ~2000.



Giallo: configurazione di default
Verde: ottimizzatore senza predizione del carico
Azzurro: ottimizzatore con predizione (DeepAR)
Blu: ottimizzatore con predizione (Prophet)



DBMS	F. obiettivo	Vincolo	Ottimizzatore	CR	F
MongoDB	min(memoria)	< 9ms latenza	Default cfg.	0	3.5%
			No predizione	71%	3%
			Prophet	61%	2.5%
			DeepAR	**76%**	**2.4%**
MongoDB	min(memoria)	< 10ms latenza	Default cfg.	0	6.0%
			No predizione	**47%**	2.7%
			Prophet	32%	2.9%
			DeepAR	45%	**1.7%**
Cassandra	min(latenza)	< 1.9 GB memoria	Default cfg.	0	0
			No predizione	**24.4%**	1.0%
			Prophet	21%	**0.7%**
			DeepAR	24%	0.8%
Cassandra	min(latenza)	< 1.9 GB memoria	Default cfg.	0	0
			No predizione	43%	0.6%
			Prophet	40%	**<0.1%**
			DeepAR	**45%**	0.1%

Conclusioni e sviluppi futuri

- Gli ottimizzatori sviluppati sono stati sempre capaci di **trovare buone configurazioni** che migliorano le performance rispetto alla configurazione di base.
- **L'utilizzo di modelli predittivi ha reso il processo di ottimizzazione più sicuro** in termini di fallimenti e violazione dei vincoli imposti sul sistema.

Sviluppi futuri:

- Usare i moduli di predizione per **applicare proattivamente la migliore configurazione** trovata per il futuro carico di lavoro.

Grazie per l’attenzione!