



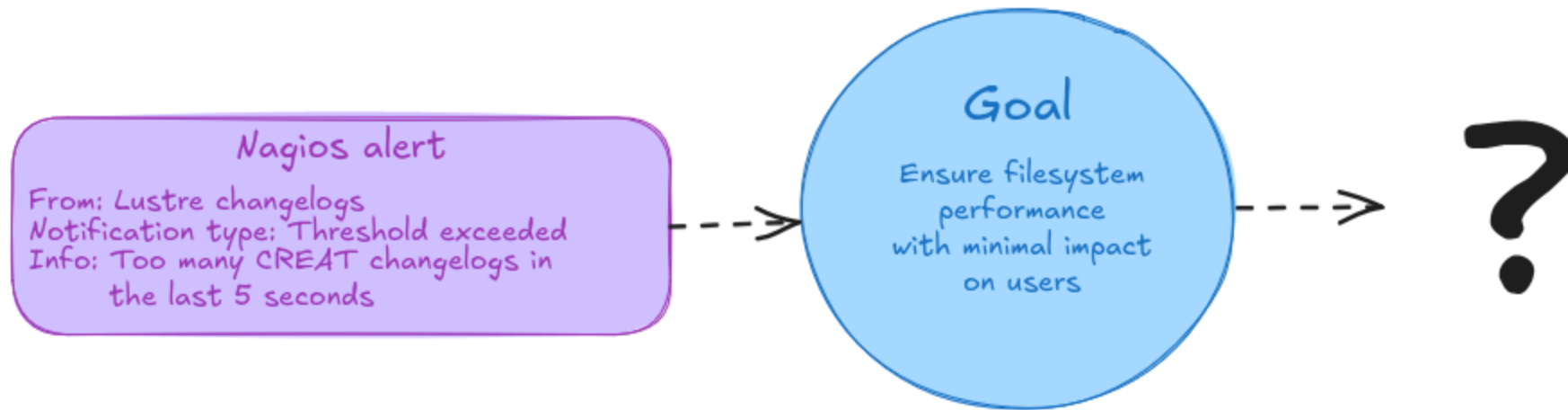
Kraken Storage Agent

An AI agent to mitigate Lustre system overusage issues

Smart HPC

Sophie Willemot, Ludovic Mustière, Laëtitia Anton, Sébastien Gougeaud

Use case: monitoring the storage system

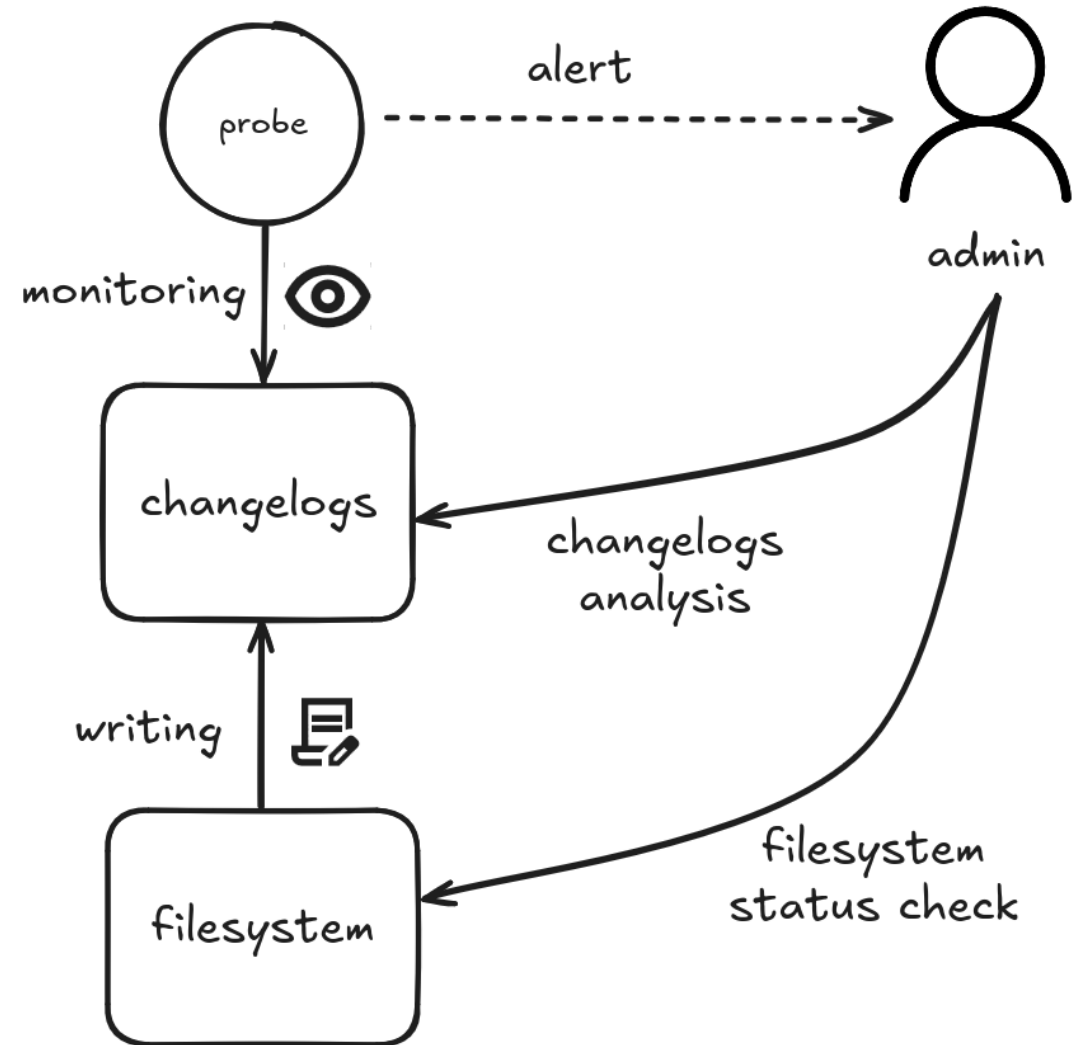


Before Kraken Storage

An administrator receives an **alert**

- need to analyse changelogs
- check the current status of the filesystem
- often **post-mortem**
- find the best course of actions and apply it

Monitoring changelogs is fastidious and time consuming



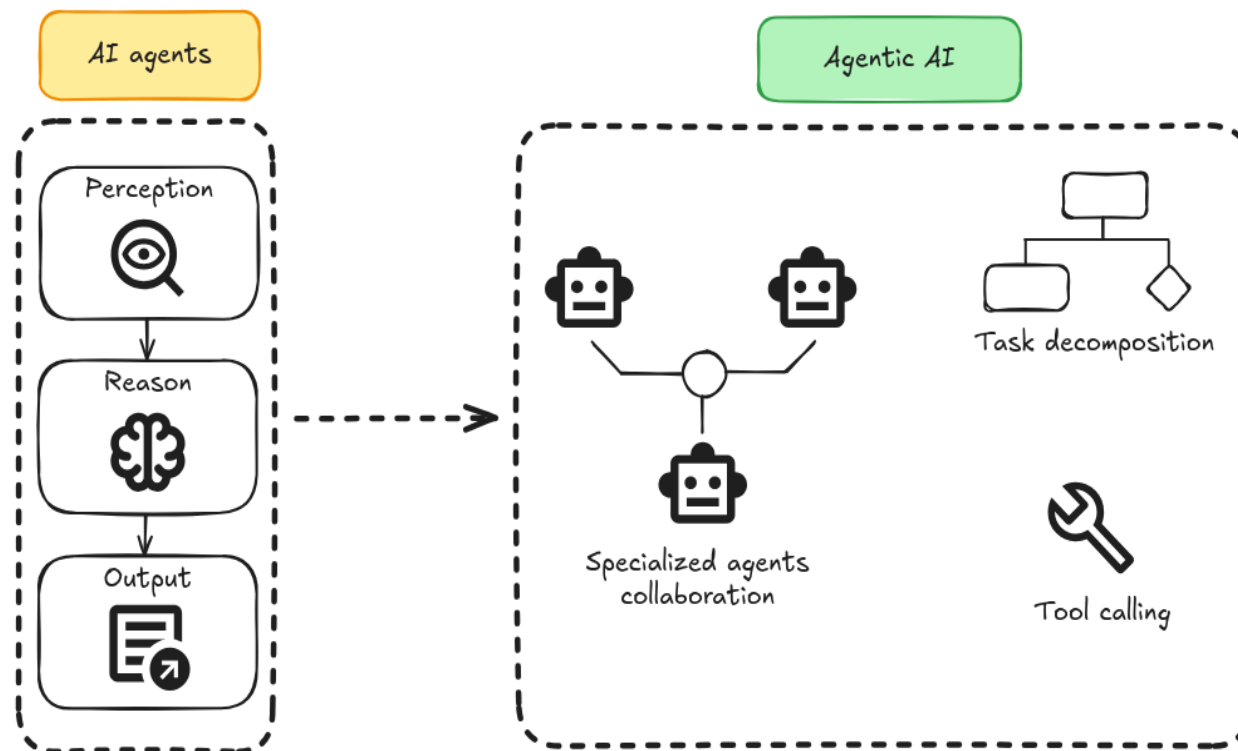
Emergence of agentic AI

Agentic AI are **autonomous agents** able to

- understand **long-term goals**
- develop advanced strategies
- execute **complex workflows**

Using **Chain of Thought (CoT)** technique

- simulate **human-like reasoning**
- accurately solve multistep problems
- attention to detail



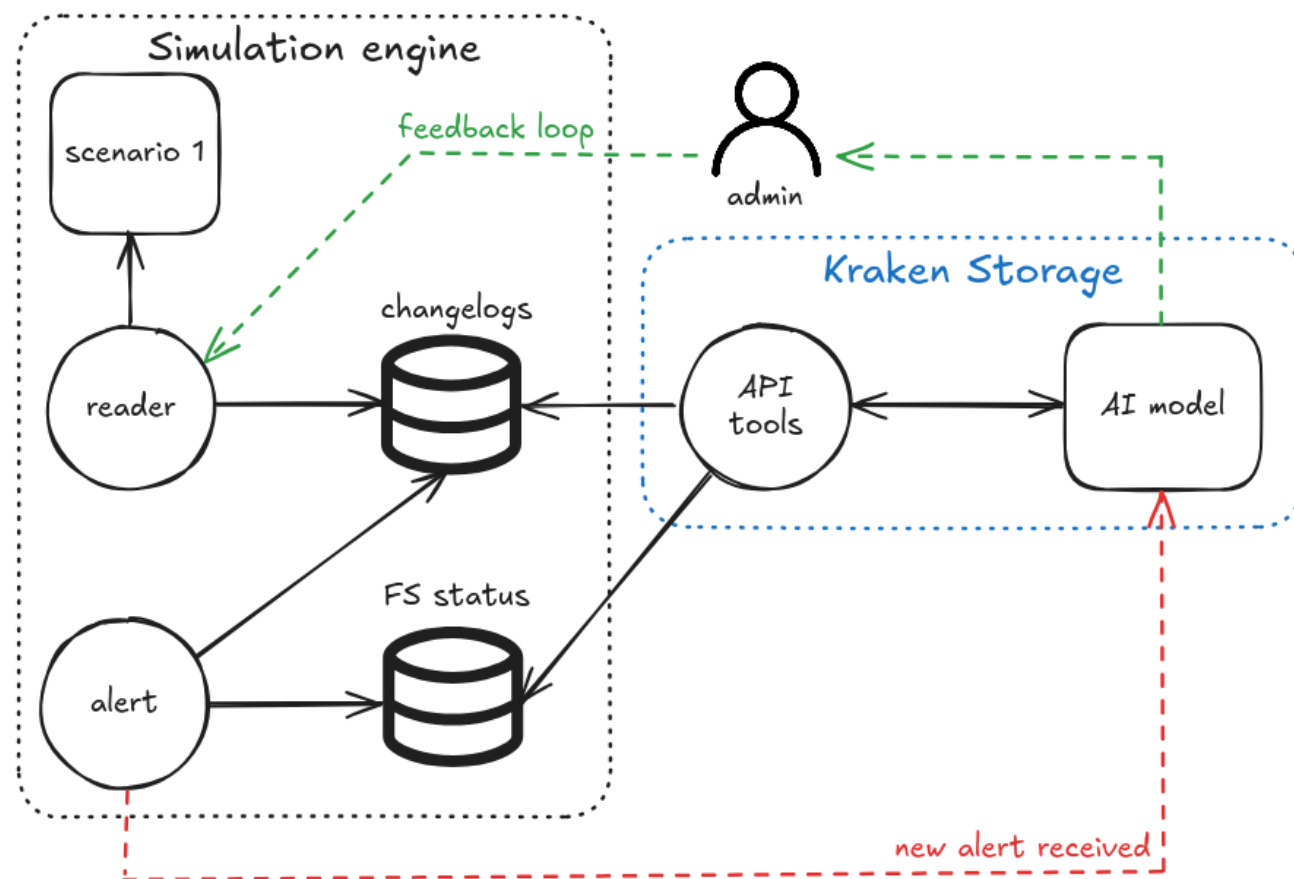
Architecture of Kraken Storage Agent

Specialized storage agent

- perception
 - previous **alerts analysis**
 - current status of the FS
- reason
 - jobs priority
 - **event types custom answer**
 - combined user changelogs
- output
 - put one or more jobs into quarantine
 - **alert the concerned users**

Implementation requirements

- data changelogs
- specialized reasoning or agentic LLM



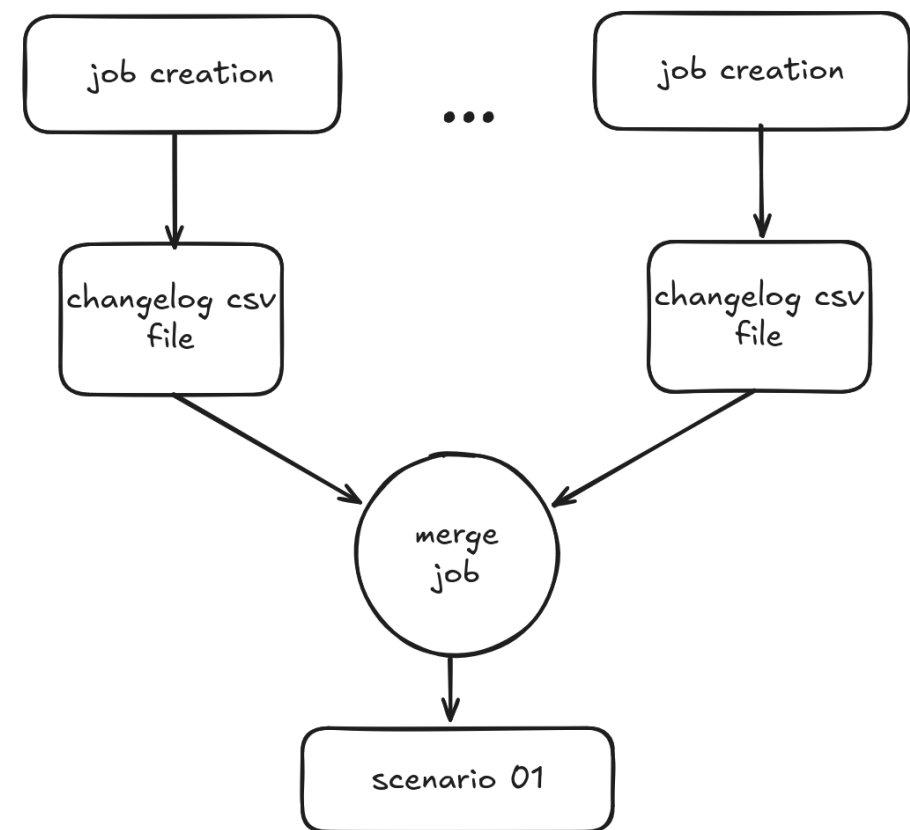
Lustre changelogs generation

Create **replayable scenarios** to train and test the model on synthetic data

- one CSV file per job
- job **timestamps** are **randomized**
- merge any number of jobs together to create custom scenarios

Parameters for a job

- user id, job id, number of occurrences, event pattern, start, end...



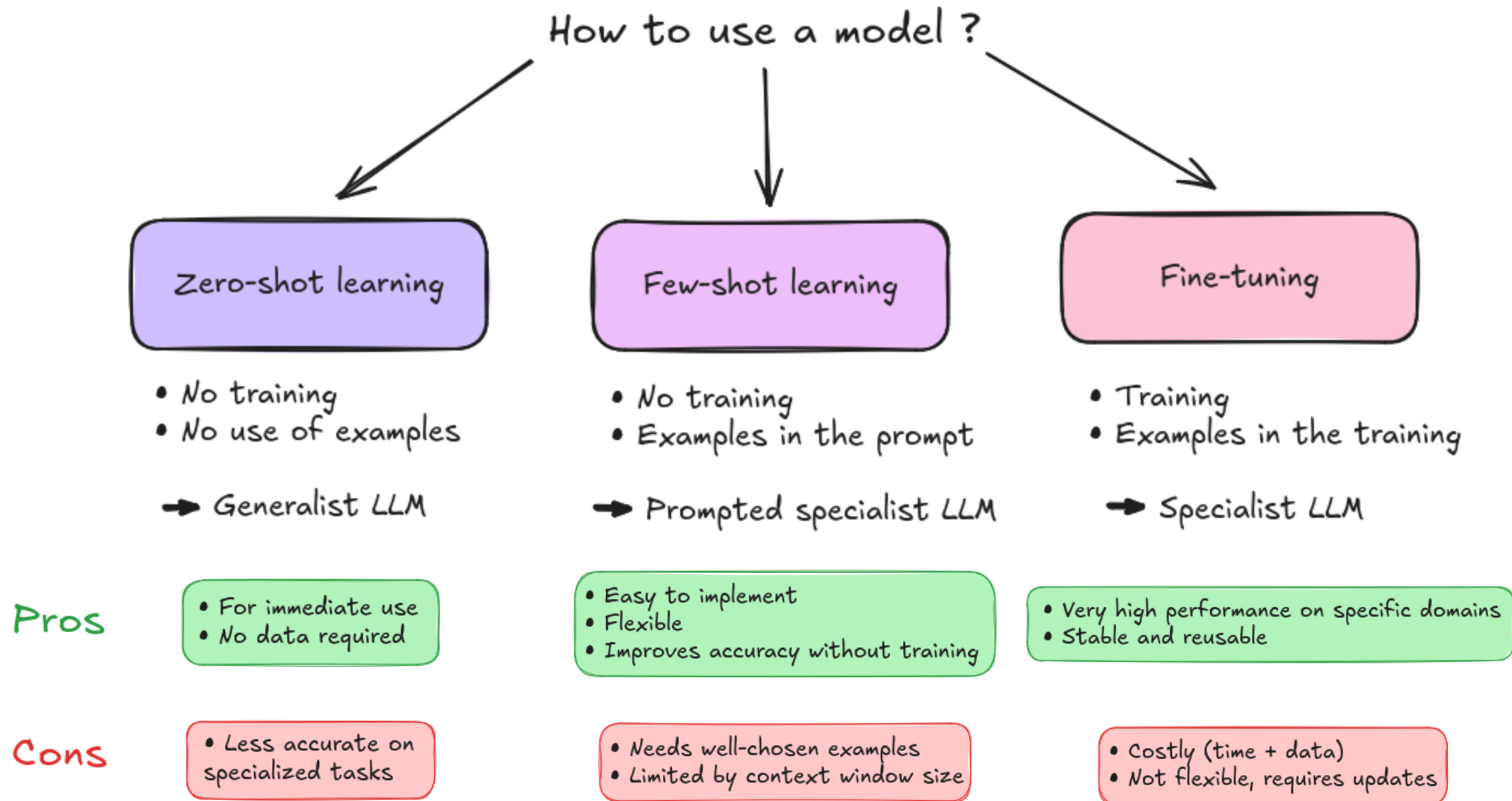
timestamp (randomized) static elements event type timestamp (randomized)

2025/06/24 08:00:03 [27346/1] lustre-MDT0000: 1000000011 01CREAT 1750752003.768512011

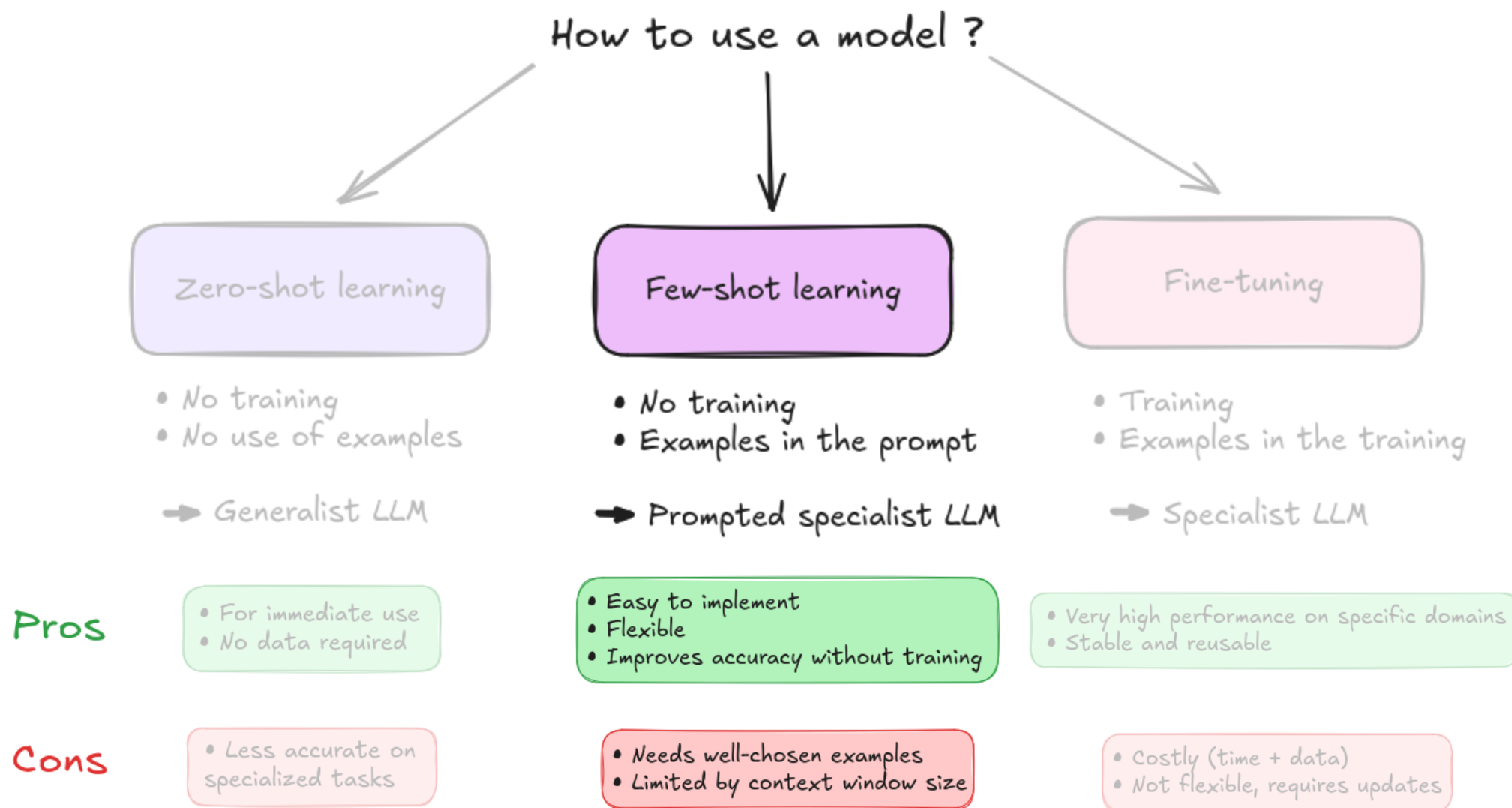
0x0 t=[0x1737946802;0x1:0x0] J=node0001.i:10000001.0001

job id user id

What to do with a LLM?



What to do with a LLM?



Methods & Performance

- Based on **Devstral Small** from Mistral
 - agentic LLM for software engineering tasks
 - 24B parameters
 - 128k context window
 - Apache 2.0 license
 - 4-bit quantization
- **5-shot learning** with custom CoT
 - 5 descriptive use cases
- **Fast inference model**
 - model loading time ~2min
 - total time to solve the problem ~45sec
 - 2 GPU NVIDIA A100 (40GB RAM)

Demo 1 – Targeted job

```
"role": "user",
"content": "J'ai reçu une alerte liée aux changelogs Lustre. Trouve moi d'où vient le problème et comment le résoudre.",
},
{
"role": "assistant",
"content": "D'accord, pour commencer j'ai besoin de regarder les précédentes alertes pour récupérer le détail de l'alerte et également détecter si d'autres alertes récentes similaires ont été remontées. Pour regarder les précédentes alertes, je dois accéder à la base de données des alertes. Il faut alors que je fasse appel à l'outil 'get last alerts' qui prend en arguments une durée en secondes. Regardons les précédentes alertes pour les 10 dernières secondes.get_last_alerts(\"seconds\": 10)\"
},
{
"role": "tool",
"name": "get last alerts",
"content": \"[{ '1750752083.951452', 'MAX_01CREAT' } ]\",
"tool_call_id": "uSp67ZjZ4\"
},
{
"role": "assistant",
"content": \"La fonction renvoie une liste qui contient des tuples avec le format (timestamp, alerte). Ici, MAX_01CREAT signifie que l'alerte reçue est liée à un trop grand nombre de CREAT. Cette alerte est importante car une trop grande occurrence peut entraîner une augmentation importante du nombre d'index, ce qui peut être dangereux pour le système de fichiers. Il faut maintenant que je compte la fréquence d'apparition de chaque opération pour savoir le nombre précis de 01CREAT. Pour cela, je fais appel à l'outil 'get types count' qui prend en arguments une durée en secondes. Regardons le nombre de chaque opération pour les 10 dernières secondes.get_types_count(\"seconds\": 10)\"
},
{
"role": "tool",
"name": "get types count",
"content": \"[{ '01CREAT': 134, '02MKDIR': 10, '07RMDIR': 11, '11CLOSE': 26 } ]\",
"tool_call_id": \"zT9XxpRdu\"
},
{
"role": "assistant",
"content": \"Je vais maintenant analyser les résultats de la fonction. Je dispose d'un dictionnaire avec en clé, chaque opération présente et en valeur sa fréquence d'apparition dans les 10 dernières secondes. Ici, on a plusieurs opérations présentes mais on s'intéresse uniquement au 01CREAT. Regardons maintenant s'il est possible de cibler un ou plusieurs jobs responsables de ces opérations 01CREAT. Pour cela, je fais appel à l'outil 'get jobs count by type' afin de récupérer la distribution des jobs responsables de l'opération 01CREAT. Cet outil prend en arguments une durée en secondes afin de récupérer la répartition sur cette période et l'opération de méta-données que l'on souhaite cibler. Ici, je vais regarder pour les 10 dernières secondes et uniquement pour l'opération 01CREAT.get_jobs_count_by_type(\"seconds\": 10, \"event_type\": \"01CREAT\")\"
},
{
"role": "tool",
"name": "get jobs count by type",
"content": \"[{ 'i10000001': 9, 'i10000007': 111, 'i10000008': 7, 'i10000010': 7 } ]\",
"tool_call_id": \"vLz58FFd\"
},
{
"role": "assistant",
"content": \"La fonction retourne un dictionnaire avec en clé les job id responsables de l'opération passée en argument et en valeur le nombre de méta-données générées par ce job. Ici, on a 4 clés, c'est-à-dire 4 jobs qui sont responsables des opérations 01CREAT. Récupérons maintenant le seuil pour l'opération 01CREAT afin de voir si les 4 jobs dépassent ce seuil, s'il n'y en a qu'un seul ou aucun qui dépasse le seuil. Pour cela, j'utilise l'outil 'get config' qui prend en argument un nom de variable dont on veut savoir le seuil. Ici, je passe en nom de variable l'alerte reçue : MAX_01CREAT.get_config(\"threshold_name\": \"MAX_01CREAT\")\"
},
{
"role": "tool",
"name": "get config",
"content": \"50\",
"tool_call_id": \"jkg6VXHxF\"
},
{
```

```
"role": "assistant",
"content": \"Le seuil est de 50, il y a donc 1 job qui dépasse le seuil : ['i10000007'] car il est responsable de 111 opérations de ce type ce qui est supérieur au seuil de 50. À l'inverse, les jobs i10000001, i10000008 et i10000010 sont responsables de seulement 9, 7 et 7 opérations respectivement ce qui est inférieur à 50, donc ces jobs ne sont pas impactants pour le système de fichiers. Ainsi, pour résoudre l'alerte, il est nécessaire de prévenir tous les utilisateurs qui ont lancé ces jobs afin d'éviter que leurs futurs jobs n'aient le même comportement. Pour récupérer l'id de chaque utilisateur pour chaque job, j'utilise l'outil 'get user by jobs' qui prend en entrée une liste d'id de job. Je passe donc en argument les id des jobs ciblés : ['i10000007'].get_user_by_jobs(\"jobs\": ['i10000007\\n'])\"
},
{
"role": "tool",
"name": "get user by jobs",
"content": \"{ 'i10000007': '0007' }\",
"tool_call_id": \"XWUebJLYH\"
},
{
"role": "assistant",
"content": \"J'ai récupéré l'identifiant de chaque utilisateur concerné, je vais maintenant préparer le mail à leur envoyer. Il doit contenir les informations sur son job, la cause de l'alerte, quelques manières d'optimiser l'utilisation des ressources et lui indiquer de nous contacter pour toute assistance supplémentaire. Voici le message pour l'utilisateur 0007 :
Objet : Notification concernant l'utilisation intensive des entrées/sorties par votre job i10000007.

Nous avons détecté que votre job i10000007 effectue un nombre élevé d'opérations de création de fichiers (01CREAT), ce qui entraîne une utilisation intensive des entrées/sorties (IO) sur notre système de fichiers. Une telle utilisation peut affecter les performances globales de notre infrastructure HPC.

Pour optimiser l'utilisation des ressources, nous vous recommandons de :

1. Regrouper les opérations d'écriture/lecture pour minimiser les accès disque.
2. Optimiser les algorithmes pour réduire la fréquence des opérations de création et d'accès aux fichiers.

Nous restons à votre disposition pour toute assistance ou conseil supplémentaire dont vous pourriez avoir besoin pour optimiser vos jobs. Merci de votre coopération.

Cordialement,
TGCC-CRRT Hotline

Résumé des actions effectuées :
Pour résoudre l'alerte, j'ai procédé de cette manière :
1. Récupération de l'alerte et des précédentes alertes existantes s'il y en a : Il n'y a qu'une seule alerte : MAX_01CREAT.
2. Identification du ou des jobs responsables de l'alerte : Il y a 1 jobs responsables : ['i10000007'].
3. Identification du ou des utilisateurs liés aux jobs. Il y a 1 utilisateurs liés aux jobs : ['0007'].
4. Rédaction d'un mail pour prévenir chaque utilisateur du comportement dangereux de son job.
```

Demo 1 – Targeted job

Thinking and tools call

User: I received an alert related to Lustre changelogs.
Please determine what the problem is and how to fix it.

Assistant: Need to review previous alerts to retrieve the details.

Tool: `get_last_alerts({'seconds': 10})`

-> `[('1750752084.6040907', '[MAX_01CREAT]')]`

Assistant: Need to get the precise number of 01CREAT.

Tool: `get_types_count({'seconds': 10})`

-> `{'01CREAT': 134, '02MKDIR': 10, '11CLOSE': 26}`

Assistant: Need to target the jobs responsible of 01CREAT.

Tool: `get_jobs_count_by_type({'seconds': 10, 'event_type': '01CREAT'})`

-> `{'i10000001': 9, 'i10000007': 111, 'i10000008': 14}`

Assistant: Need to retrieve the threshold for the 01CREAT operation.

Tool: `get_jobs_count_by_type({'threshold_name': 'MAX_01CREAT'})`

-> 50

Assistant: Some jobs exceeded the threshold. Need to retrieve the ID of each user for each job.

Tool: `get_user_by_jobs({'jobs': ['i10000007']})`

-> `{'i10000007': '0007'}`

Final results

Assistant: Preparing email to be send to the user.
The email should contain information about the user's job, the cause of the alert, ways to optimize resource utilization, and instructions on how to contact us for further assistance.
Below is the message for user 0007: [...]

Summary of actions taken:

To resolve the alert, I proceeded as follows:

1. Retrieve the alert and any existing alerts:
There is only one alert, MAX_01CREAT.
2. Identify the job(s) responsible for the alert:
One job is responsible: i10000007.
3. Identify the user(s) linked to the jobs:
One user is linked to the job: [0007].
4. Send an email to the user to warn him about the dangerous behavior of his job.

Demo 2 – Targeted user

Thinking and tools call

User: I received an alert related to Lustre changelogs.
Please determine what the problem is and how to fix it.

Assistant: Need to review previous alerts to retrieve the details.

Tool: `get_last_alerts({'seconds': 10})`
-> `[('1750752100.0', '[MAX_02MKDIR]')]`

Assistant: Need to get the precise number of 02MKDIR.

Tool: `get_types_count({'seconds': 10})`
-> `{'01CREAT': 11, '02MKDIR': 112, '07RMDIR': 4}`

Assistant: Need to target the jobs responsible of 02MKDIR.

Tool: `get_jobs_count_by_type({'seconds': 10, 'event_type': '02MKDIR'})`
-> `{'i10000056': 35, 'i10000083': 46, 'i10000072': 31}`

Assistant: Need to retrieve the threshold for the 02MKDIR operation.

Tool: `get_jobs_count_by_type({'threshold_name': 'MAX_02MKDIR'})`
-> 50

Assistant: **No job exceeds the threshold. Try to target some users.**

Tool: `get_user_by_jobs({'jobs': ['i10000056', 'i10000083', 'i10000072']})`
-> `{'i10000056': '0041', 'i10000083': '0045', 'i10000072': '0045'}`

Final results

Assistant: **A user is targeted because the sum of the operations of these jobs is greater than the threshold.** Preparing emails to be send to the users. Each email should contain information about the user's job, the cause of the alert, ways to optimize resource utilization, and instructions on how to contact us for further assistance.
Below is the message for user 0045: [...]

Summary of actions taken:

To resolve the alert, I proceeded as follows:

1. Retrieve the alert and any existing alerts:
There is only one alert, MAX_02MKDIR.
2. Identify the job(s) responsible for the alert:
No responsible job.
3. Identify the user(s) linked to the jobs:
One user is linked to the jobs: [0045].
4. Send an email to the user to warn him about the dangerous behavior of their jobs.

Demo 3 – Quarantined jobs

Thinking and tools call

User: I received an alert related to Lustre changelogs.
Please determine what the problem is and how to fix it.

Assistant: Need to review previous alerts to retrieve the details.

Tool: `get_last_alerts({'seconds': 10})`
-> `[('1750752010.685263 ', '[MAX_01CREAT]')]`

Assistant: Need to get the precise number of 01CREAT.

Tool: `get_types_count({'seconds': 10})`
-> `{'01CREAT': 50}`

Assistant: Need to target the jobs responsible of 01CREAT.

Tool: `get_jobs_count_by_type({'seconds': 10, 'event_type': '01CREAT'})`
-> `{'i100000100': 15, 'i10000091': 21, 'i10000092': 9, 'i10000093': 5}`

Assistant: Need to retrieve the threshold for the 01CREAT operation.

Tool: `get_jobs_count_by_type({'threshold_name': 'MAX_01CREAT'})`
-> 50

Assistant: Need to retrieve the ID of each user for each job.

Tool: `get_user_by_jobs({'jobs': ['i10000001', 'i10000007']})`
-> `{'i100000100': '0010', 'i10000091': '0001', 'i10000092': '0002', 'i10000093': '0003'}`

Assistant: Impossible to target a user. Jobs must be quarantined until the alert is resolved.

Tool: `ban_jobs_by_id({'jobs': ['i10000091']})`

Final results

Assistant: The jobs have been successfully quarantined.

Summary of actions taken:

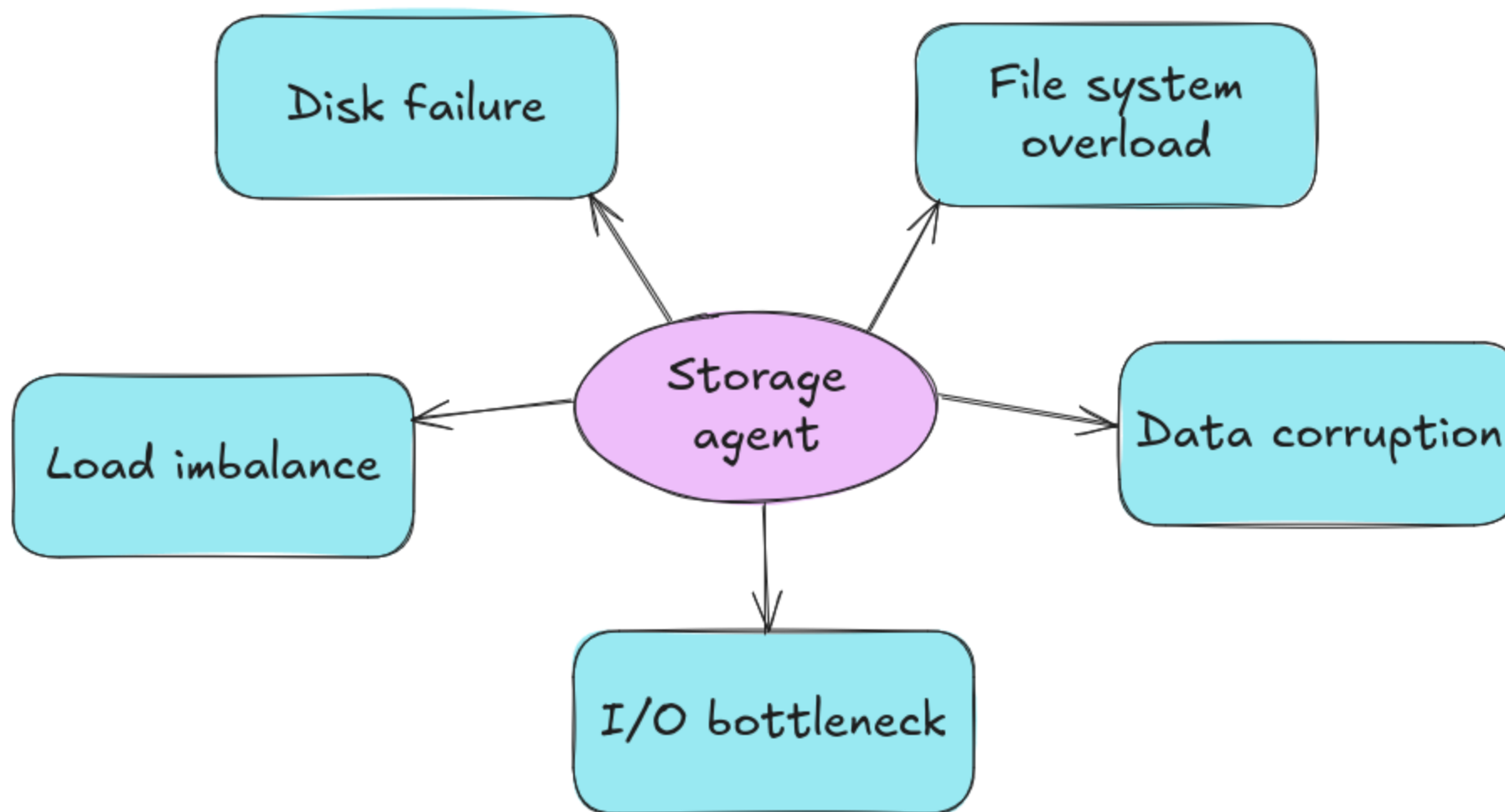
To resolve the alert, I proceeded as follows:

1. Retrieve the alert and any existing alerts:
There is only one alert: MAX_01CREAT.
2. Identify the job(s) responsible for the alert:
No responsible job.
3. Identify the user(s) linked to the jobs:
No users are linked to the jobs.
4. **Quarantine job(s)** responsible for operations until they fall below the threshold:
One job has been quarantined: i10000091.

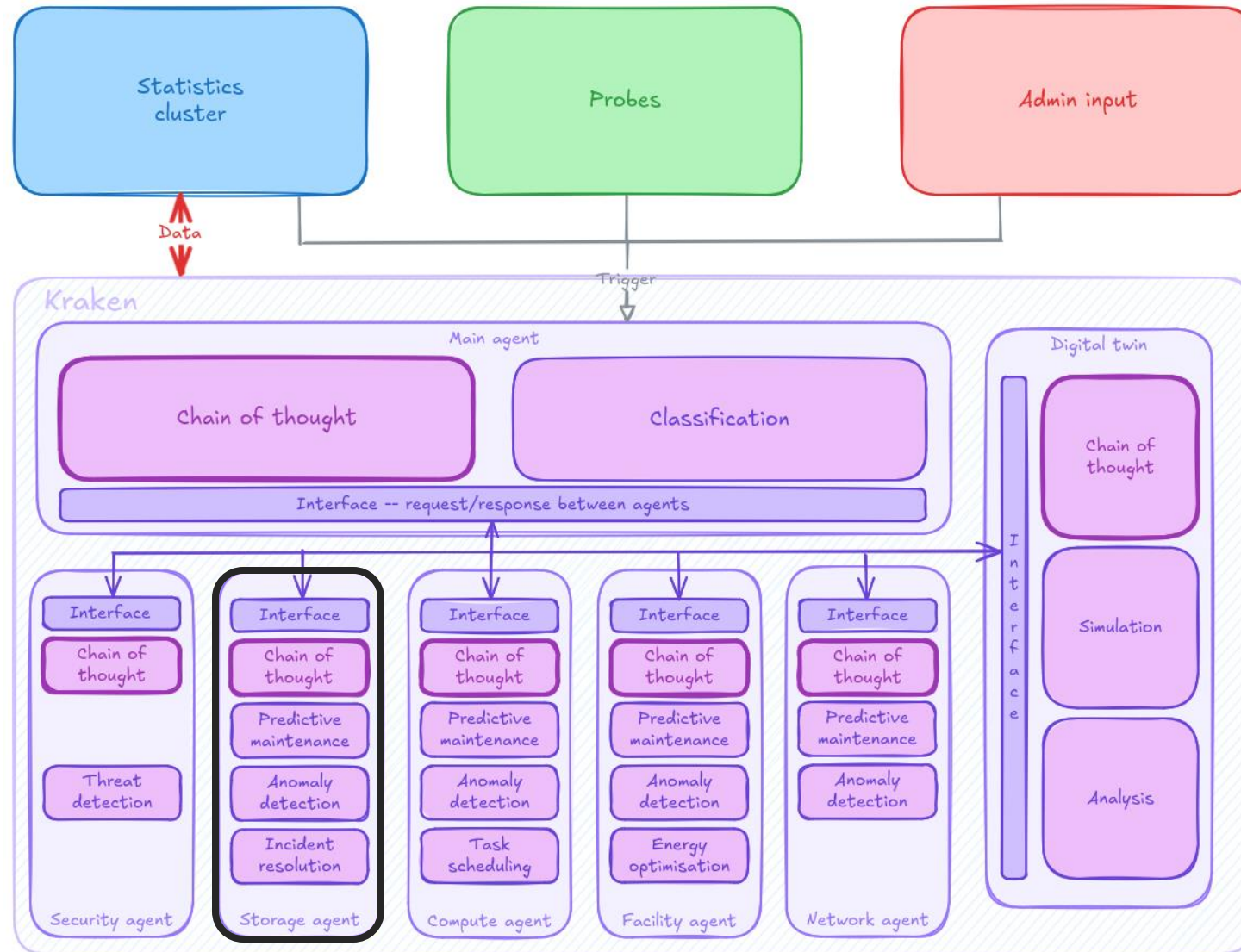
How to improve our agent

- **Fine-tuning the model** for improved performance
 - Low-Rank Adaptation (LoRA) training
 - Unsloth framework
- Diversifying scenarios to enhance robustness
- Expanding tool-calling with additional integrations
- Leveraging real-world data for validation and scalability

Beyond Lustre changelogs

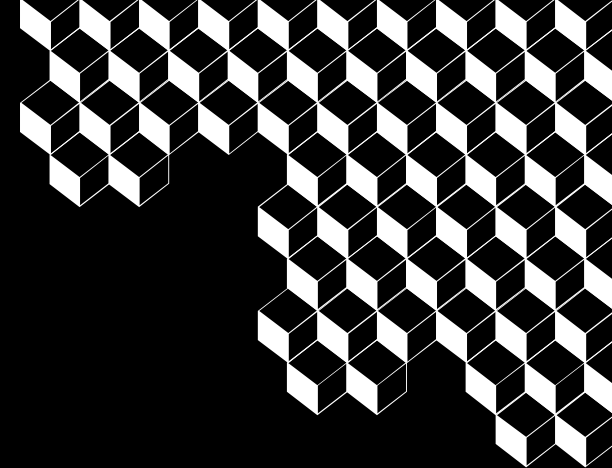


Kraken



Conclusion

- Our storage agent is able to mitigate Lustre system overusage issues
 - specialized tool-calling agent
 - few-shot learning using 5 examples in the prompt
 - simple scenarios can be resolved under a minute
- Improvements can be made
 - fine-tuning with new scenarios
 - use of real data
 - improve the tool API
- Our storage agent is part of Kraken, our agentic AI that aims to help managing the entire HPC computing center
 - a main agent will decompose and distribute the tasks to be solved
 - one or more specialized agent per computing center component



Questions?

sophie.willemot@cea.fr

ludovic.mustiere2@cea.fr

sebastien.gougeaud@cea.fr

thomas.leibovici@cea.fr