A propos Logs/Traces/Metrics

Métriques

Les métriques sont des mesures quantitatives permettant d'évaluer le comportement et la performance d'un système au fil du temps .

- Données numériques
- Séries temporelles
- Structure définie

- Taux d'utilisation du processeur : 45 %
- Mémoire utilisée : 2,5 Go
- Temps de réponse moyen : 200 ms
- Nombre de requêtes traitées : 50 requêtes/seconde

Logs

Les logs sont des enregistrements d'événements .Ils fournissent des informations détaillées sur ce qui se passe à un instant précis, notamment en cas d'erreurs, d'avertissements ou d'activités spécifiques.

- Horodatés
- Riches en contexte
- Peu ou non structurés

```
[2024-09-17 12:35:22] INFO: User 'john.doe' requested '/login' page.
Response time: 45ms
[2024-09-17 12:35:25] ERROR: Authentication failed for user 'john.doe'.
Invalid credentials.
```

Traces

Les traces permettent de suivre le parcours complet d'une requête au sein d'un système distribué.

Offrent une représentation structurée ou visuelle de ce cheminement

- Centrées sur la requête
- Composées de spans
- Vision distribuée

```
Trace ID: 12345
- Span 1 (Frontend Service): 30ms
- Span 2 (Authentication Service): 15ms
- Span 3 (Payment Gateway Service): 80ms
- Span 4 (Inventory Management Service): 25ms
Total time: 150ms
```

Problématique :

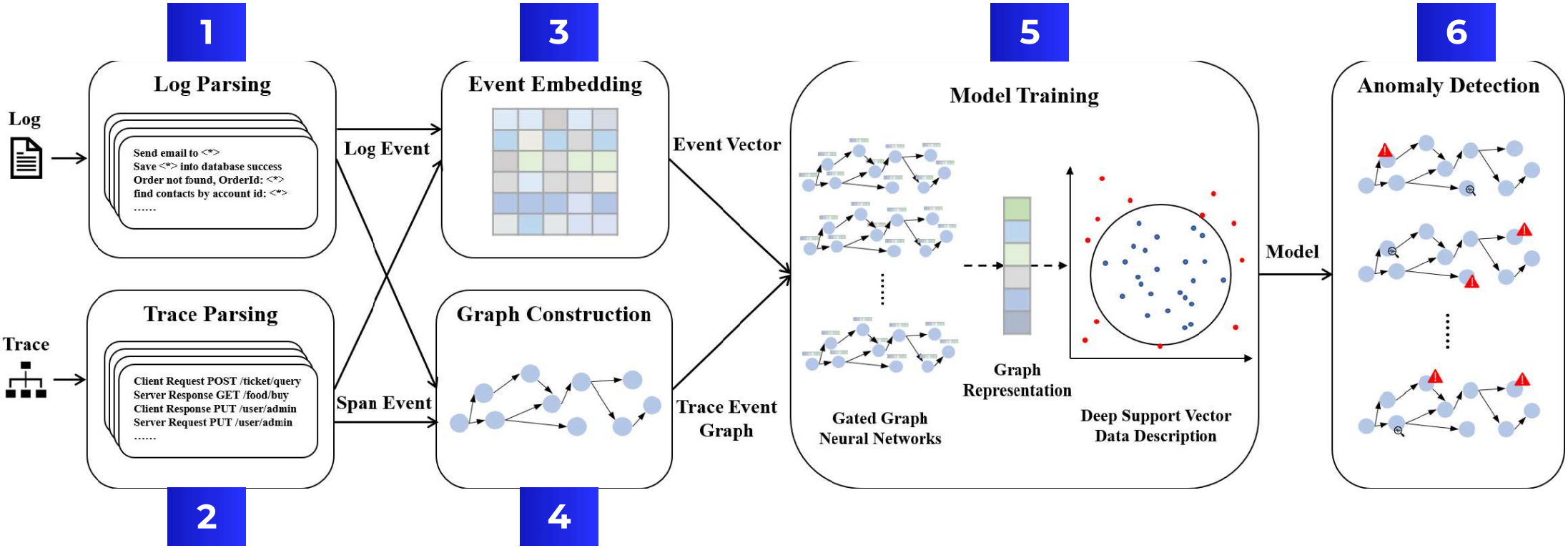
DATADOG new relic. et dynatrace sont largement utilisées dans l'industrie pour superviser les systèmes distribués permettent :

- la collecte
- l'agrégation
- la visualisation
- surveiller les performances
- détecter les anomalies

Cependant, malgré leur puissance, ces solutions présentent plusieurs limites :

- Approches principalement basées sur des règles statiques ou des seuils (CPU > 80%)
- Manque d'intelligence contextuelle (comportement normal des services|structure interne des requêtes)
- Analyse limitée des séquences ou structures profondes (anomalies complexes)

Architecture DeepTraLog :



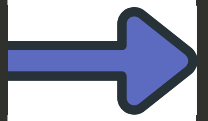
1 - Log Parsing

- Algorithme Drain
- Extraction de templates
- Extraction Paramètres dynamiques

Objectif :

Structurer les logs non structurés en modèles d'événements

```
Logs bruts d'entrée :
- "2024-01-15 10:30:12 INFO User alice logged in from IP
192.168.1.10"
- "2024-01-15 10:30:45 INFO User bob logged in from IP
192.168.1.15"
- "2024-01-15 10:31:02 ERROR User charlie failed login from IP
192.168.1.20"
- "2024-01-15 10:31:15 INFO User alice logged out"
```



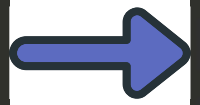
```
Template 1: "User * logged in from IP *"
  - Événement 1: (alice, 192.168.1.10, 10:30:12)
  - Événement 2: (bob, 192.168.1.15, 10:30:45)

Template 2: "User * failed login from IP *"
  - Événement 3: (charlie, 192.168.1.20, 10:31:02)

Template 3: "User * logged out"
  - Événement 4: (alice, 10:31:15)
```

Après parsing Drain

2 - Trace Parsing

- **Transformation :** Chaque span devient plusieurs événements horodatés
- **Types d'événements :** Appels synchrones : (client_request, server_receive, server_response, client_response) , Appels asynchrones : producer, consumer
- **Enrichissement :** Association avec trace ID et timestamps

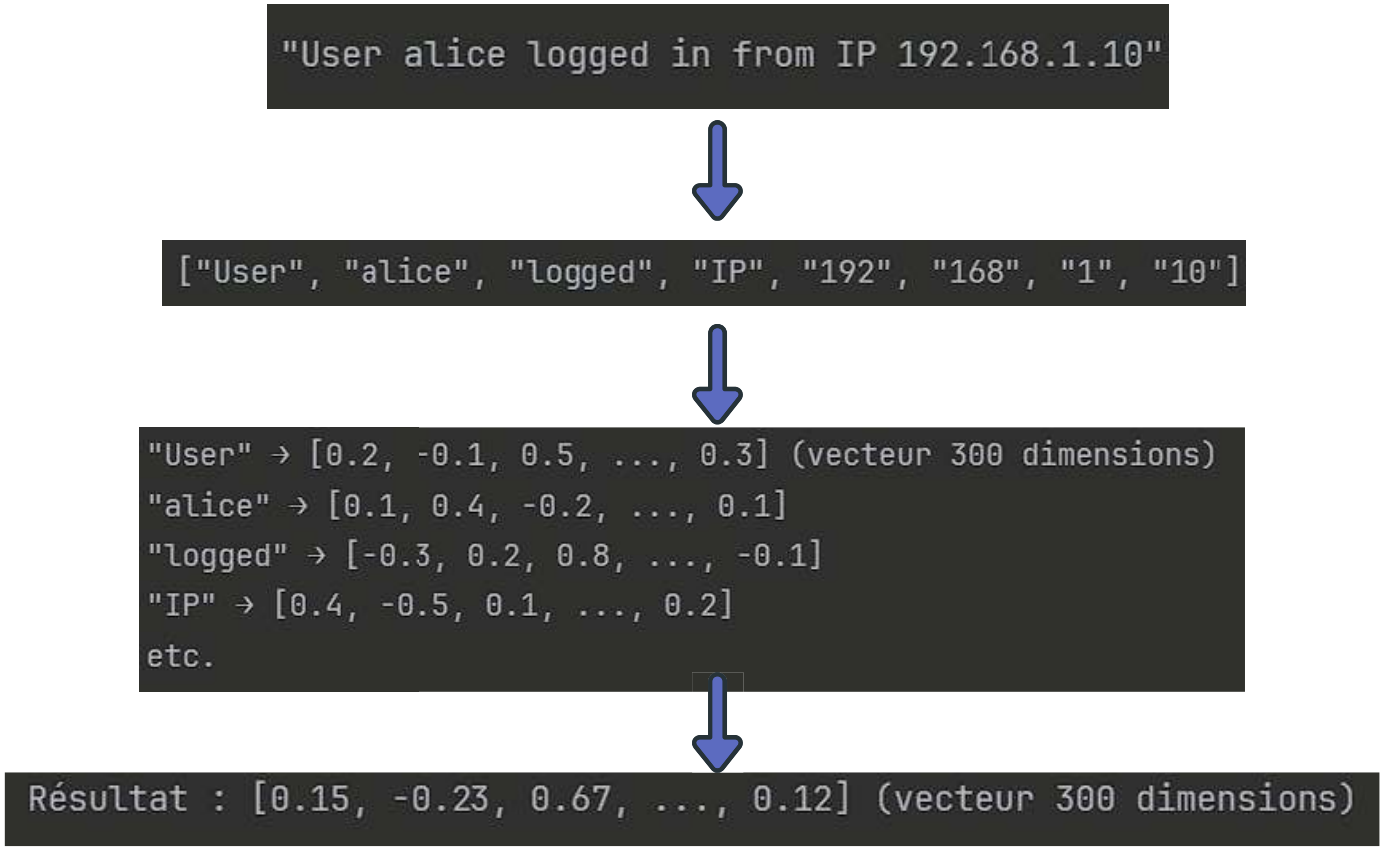
```
Trace ID: trace_abc123
Service: Frontend → Backend → Database

Span 1 (Frontend):
  - Début: 10:30:12.100, Fin: 10:30:12.800
  - Opération: "GET /api/user/profile"

Span 2 (Backend):
  - Début: 10:30:12.150, Fin: 10:30:12.750
  - Opération: "fetch_user_data"
```



```
Transformation en événements :
1. client_request (Frontend → Backend, 10:30:12.150)
2. server_receive (Backend reçoit, 10:30:12.150)
3. client_request (Backend → Database, 10:30:12.200)
```

3- Event Embedding :

- **Nettoyage :** Suppression des symboles, Suppression des stop words,Tokenisation .
- **Word Embedding (GloVe 300d) .**
- **Calcul TF-IDF .**
- **Sentence Embedding.**

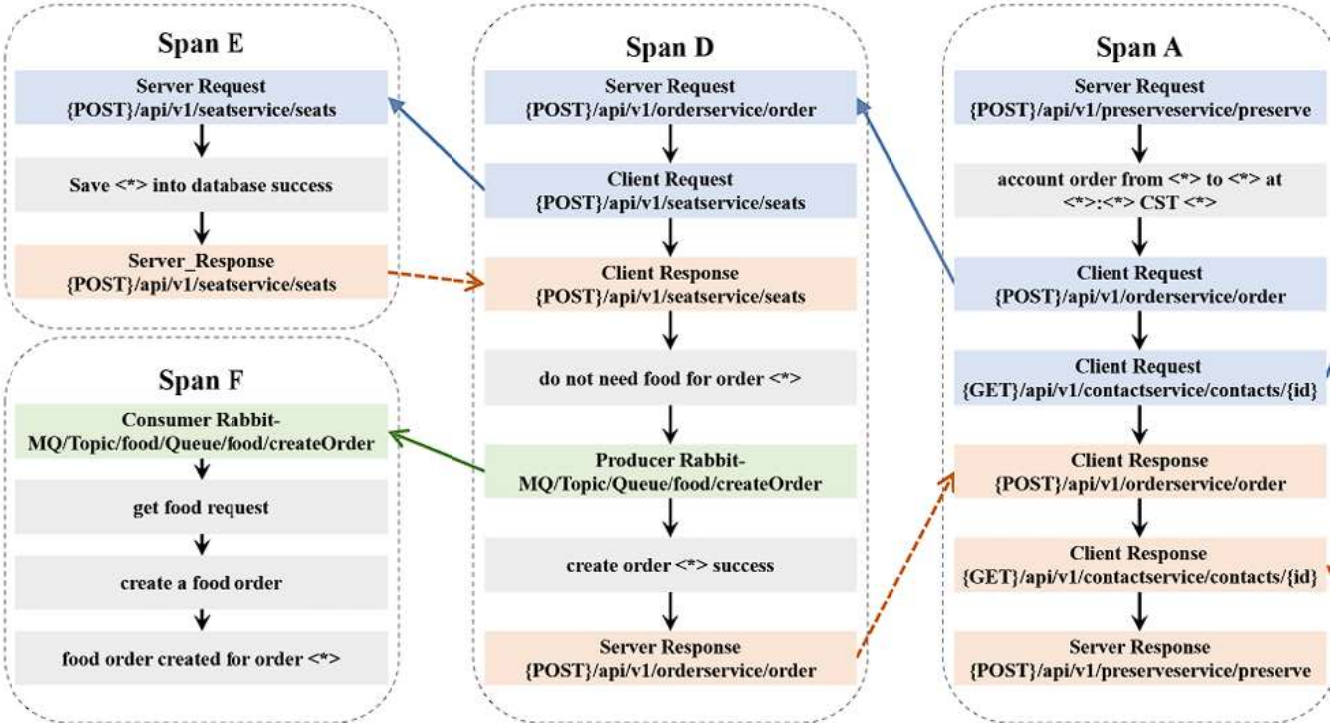


4 - Construction Graph (Trace Event Graph (TEG) :

- **Nœuds :** Événements de log et de span
- **Arêtes :** Relations temporelles et logiques entre événements

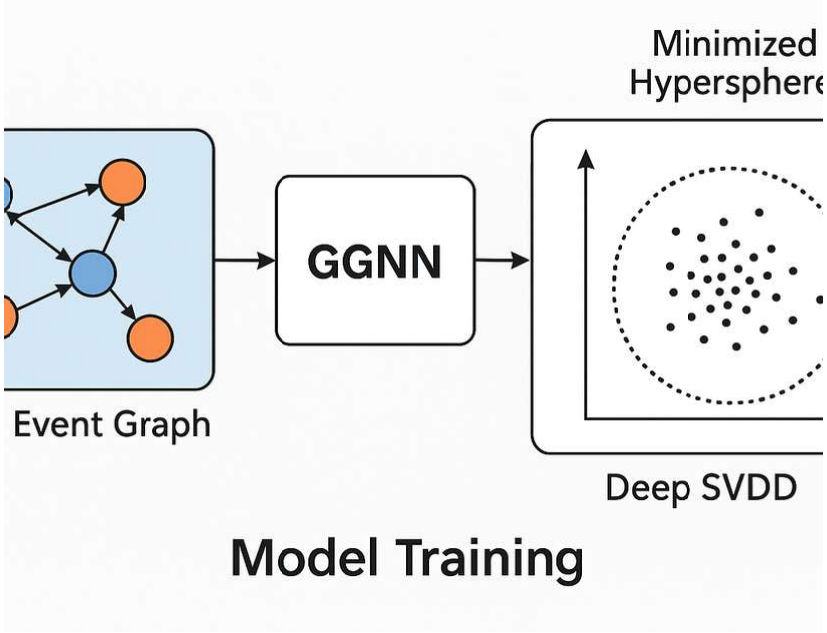
(Relations de séquence (ordre temporel) , Relations d'appel synchrone (requête/réponse) , Relations d'appel asynchrone (producteur/consommateur)

- **Avantage : Capture la structure hiérarchique et temporelle des traces**



5 - Model Training :

- **One-class-classification problem :** la majorité du bases de données est normale
- **Modèle GGNN (Gated Graph Neural Networks) :** Prend en entrée un graphe TEG (nœuds = événements, arêtes = relations de trace) / Produit une représentation vectorielle (embedding) du graphe
- **Modèle Deep SVDD (Support Vector Data Description) :** Prend l'embedding généré par GGNN , Apprend une hypersphère minimale englobant les traces normales



6 - Anomaly Detection :

- **Trace normale → Vecteur proche du centre de l'hypersphère → Classée comme normale**
- **Trace anormale → Vecteur éloigné du centre → Détectée comme anomalie 🚨**

Types d'Anomalies Détectées

- Structures de trace inhabituelles
- Séquences d'événements anormales
- Comportements temporels étranges
- Combinaisons d'événements inattendues