Documentation Complète du Solver Poker

Table des matières

Introduction

Présentation du solver poker

Le solver poker est un outil de simulation avancé conçu pour les joueurs de poker professionnels et semi-professionnels, permettant d'optimiser les décisions stratégiques basées sur des simulations massives. Il se distingue par sa capacité à simuler des milliards de mains préflop en tenant compte du modèle ICM (Independent Chip Model) et des millions de mains postflop, le tout en exploitant la puissance de calcul de plusieurs machines en mode cluster.

Contrairement aux solvers traditionnels qui fonctionnent sur une seule machine et sont limités dans leur capacité de calcul, notre solver peut distribuer les simulations sur plusieurs ordinateurs, qu'ils soient locaux ou distants, permettant ainsi d'obtenir des résultats plus précis dans des délais raisonnables.

Objectifs et avantages

Les principaux objectifs du solver poker sont :

1. **Optimisation des décisions préflop avec ICM** : Déterminer les meilleures stratégies préflop en tenant compte de la valeur réelle des jetons dans un contexte de tournoi.
2. **Analyse postflop approfondie** : Explorer les lignes de jeu optimales après le flop, le turn et la river.
3. **Calcul distribué haute performance** : Réduire considérablement le temps de calcul en exploitant plusieurs machines en parallèle.
4. **Précision et fiabilité** : Garantir des résultats statistiquement significatifs grâce au volume massif de simulations.

Les avantages par rapport aux solutions existantes incluent :

- **Vitesse de calcul supérieure** : Les simulations qui prendraient des jours sur une seule machine peuvent être réalisées en quelques heures.
- **Précision accrue** : Le nombre élevé de simulations réduit la variance et augmente la fiabilité des résultats.
- **Flexibilité** : Possibilité d'adapter les ressources de calcul en fonction des besoins.
- **Évolutivité** : Capacité à ajouter ou retirer des nœuds de calcul dynamiquement.

Public cible

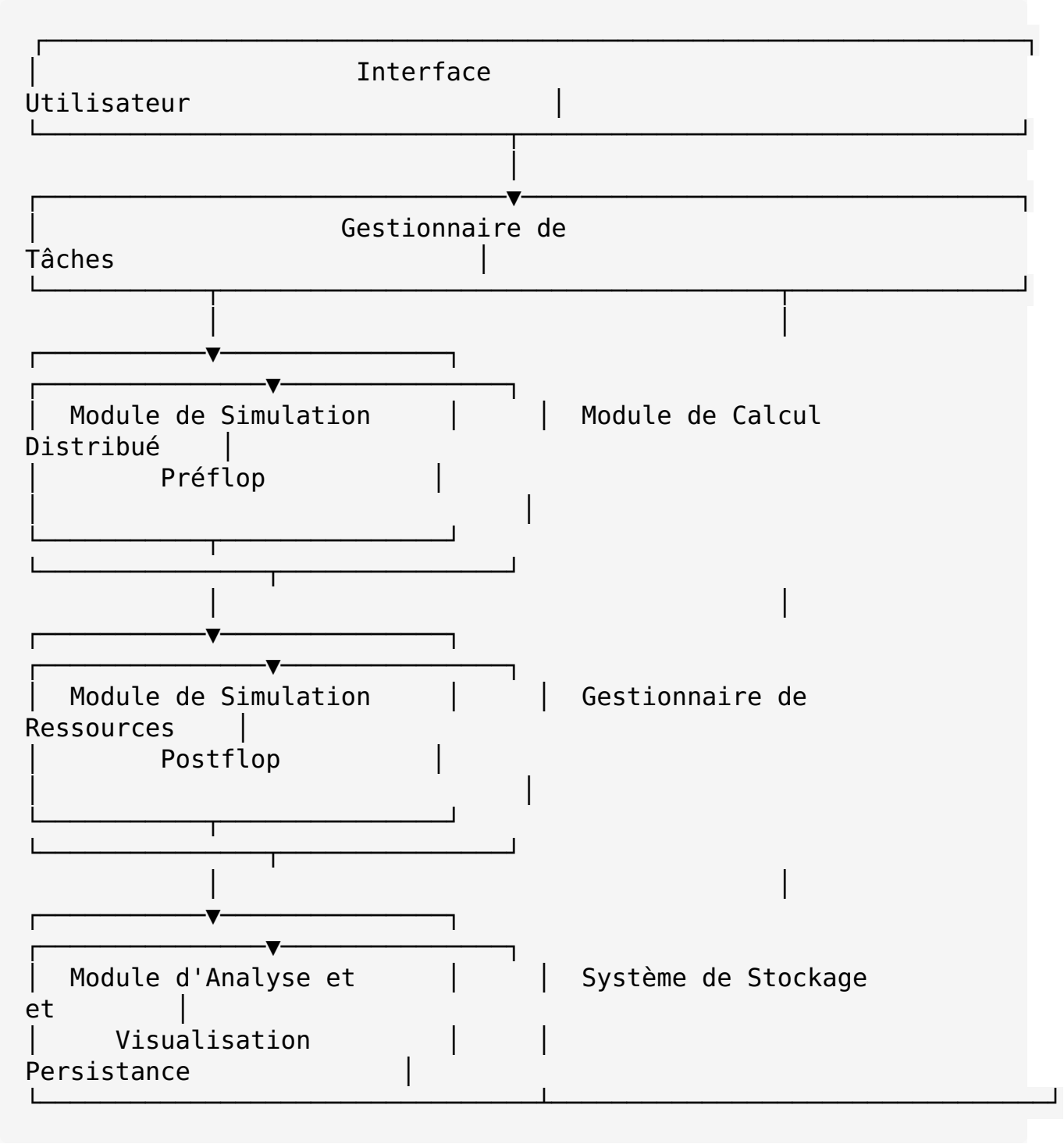
Ce solver s'adresse principalement à :

- **Joueurs professionnels** : Pour l'analyse approfondie des stratégies et l'optimisation du jeu.
- **Coachs et formateurs** : Pour illustrer des concepts avancés et créer du matériel pédagogique.
- **Équipes de poker** : Pour le développement collaboratif de stratégies.
- **Chercheurs** : Pour l'exploration de la théorie des jeux appliquée au poker.
- **Développeurs de bots** : Pour l'entraînement et l'évaluation d'agents de poker.

Vue d'ensemble du système

Architecture globale

Le solver poker est organisé en plusieurs couches et modules interconnectés :



Cette architecture modulaire permet une séparation claire des responsabilités et facilite l'évolution indépendante de chaque composant.

Composants principaux

1. **Interface Utilisateur** : Point d'entrée pour configurer les simulations, visualiser les résultats et gérer les ressources.

2. **Gestionnaire de Tâches** : Coordonne les différentes simulations et leur distribution sur les ressources disponibles.

3. **Module de Simulation Préflop** : Spécialisé dans la simulation massive de mains préflop avec intégration de l'ICM.
4. **Module de Simulation Postflop** : Gère la simulation et l'analyse des situations postflop.
5. **Module de Calcul Distribué** : Responsable de la distribution et de la parallélisation des calculs sur plusieurs machines.
6. **Gestionnaire de Ressources** : Surveille et optimise l'utilisation des ressources de calcul disponibles.
7. **Système de Stockage et Persistance** : Gère le stockage des données de simulation, des résultats et des configurations.

Flux de données

Le flux de données typique dans le système est le suivant :

1. L'utilisateur configure une simulation via l'interface utilisateur.
2. Le gestionnaire de tâches décompose la simulation en sous-tâches indépendantes.
3. Le module de calcul distribué alloue les ressources nécessaires et distribue les tâches.
4. Les modules de simulation exécutent les calculs sur les nœuds de calcul.
5. Les résultats sont agrégés et stockés par le système de persistance.
6. Le module d'analyse traite les résultats pour extraire des insights.
7. L'interface utilisateur présente les résultats à l'utilisateur sous forme de visualisations.

Installation et configuration

Prérequis système

Matériel recommandé

Pour le nœud maître : - CPU : Processeur multi-cœurs (8+ cœurs recommandés) - RAM : 16 Go minimum, 32 Go ou plus recommandé - Stockage : SSD rapide avec au moins 100 Go d'espace libre - Réseau : Interface réseau gigabit, connexion Internet stable

Pour les nœuds de calcul : - CPU : Processeurs puissants avec de nombreux cœurs (Intel i9, AMD Ryzen 9, ou équivalent) - RAM : 16 Go minimum par nœud, 64 Go ou plus pour les simulations massives - Stockage : SSD pour les données temporaires - GPU (optionnel) : NVIDIA RTX ou Tesla pour l'accélération de certains calculs

Logiciels requis

- Système d'exploitation : Linux recommandé (Ubuntu 20.04+ ou CentOS 8+), Windows 10/11 ou macOS supportés
- Python 3.8+ avec les bibliothèques nécessaires
- Docker (recommandé pour le déploiement)
- Base de données (PostgreSQL, MongoDB)
- Système de file d'attente (Redis, RabbitMQ)

Installation du nœud maître

Installation automatisée

La méthode la plus simple pour installer le nœud maître est d'utiliser le script d'installation automatisé :

```
# Télécharger le script d'installation
curl -O https://poker-solver.example.com/install/master_install.sh

# Rendre le script exécutable
chmod +x master_install.sh

# Exécuter le script d'installation
./master_install.sh
```

Le script vous guidera à travers le processus d'installation et vous demandera les informations nécessaires pour configurer le nœud maître.

Installation manuelle

Si vous préférez une installation manuelle, suivez ces étapes :

1. Cloner le dépôt :

```
git clone https://github.com/poker-solver/poker-solver.git
cd poker-solver
```

1. Installer les dépendances :

```
# Mise à jour du système
sudo apt update && sudo apt upgrade -y

# Installation des dépendances système
sudo apt install -y build-essential python3-dev python3-pip \
```

```
    postgresql postgresql-contrib redis-server nginx

# Installation des dépendances Python
pip3 install -r requirements.txt
```

1. Configurer la base de données :

```
# Création de la base de données
sudo -u postgres psql -c "CREATE DATABASE poker_solver;"
sudo -u postgres psql -c "CREATE USER poker_user WITH PASSWORD
'secure_password';"
sudo -u postgres psql -c "GRANT ALL PRIVILEGES ON DATABASE
poker_solver TO poker_user;"

# Initialisation du schéma
python3 scripts/init_database.py
```

1. Configurer le serveur :

```
# Configuration du serveur Redis
sudo cp config/redis.conf /etc/redis/redis.conf
sudo systemctl restart redis

# Configuration du serveur API
cp config/api_server.conf.example config/api_server.conf
# Éditer le fichier de configuration avec les paramètres
appropriés
nano config/api_server.conf
```

1. Démarrer les services :

```
# Démarrage du gestionnaire de tâches
python3 master_node.py --config config/master_config.json

# Configuration du service systemd pour le démarrage automatique
sudo cp config/poker_solver_master.service /etc/systemd/system/
sudo systemctl enable poker_solver_master
sudo systemctl start poker_solver_master
```

Installation des nœuds de calcul

Installation automatisée

Pour installer un nœud de calcul de manière automatisée :

```
# Télécharger le script d'installation
curl -O https://poker-solver.example.com/install/
compute_install.sh

# Rendre le script exécutable
chmod +x compute_install.sh

# Exécuter le script d'installation
./compute_install.sh --master-uri http://master-node:8080
```

Installation manuelle

Pour une installation manuelle d'un nœud de calcul :

1. Cloner le dépôt :

```
git clone https://github.com/poker-solver/poker-solver.git
cd poker-solver
```

1. Installer les dépendances :

```
# Mise à jour du système
sudo apt update && sudo apt upgrade -y

# Installation des dépendances système
sudo apt install -y build-essential python3-dev python3-pip

# Installation des dépendances Python
pip3 install -r requirements.txt
```

1. Configurer l'agent de calcul :

```
# Copie du fichier de configuration exemple
cp config/compute_node.conf.example config/compute_node.conf

# Édition du fichier de configuration
nano config/compute_node.conf
```

1. Démarrer l'agent :

```
# Démarrage manuel de l'agent
python3 compute_node.py --config config/compute_node.conf

# Configuration du service systemd pour le démarrage automatique
sudo cp config/poker_solver_compute.service /etc/systemd/system/
```

```
sudo systemctl enable poker_solver_compute
sudo systemctl start poker_solver_compute
```

Configuration initiale

Après l'installation, vous devez effectuer une configuration initiale du système :

1. Accéder à l'interface web d'administration :

```
http://master-node:8080/admin
```

1. Se connecter avec les identifiants par défaut :

```
Utilisateur : admin
Mot de passe : poker_solver_admin
```

1. Changer le mot de passe administrateur :
2. Aller dans "Paramètres" > "Utilisateurs"
3. Sélectionner l'utilisateur "admin"
4. Cliquer sur "Modifier" et changer le mot de passe
5. Configurer les paramètres globaux :
6. Aller dans "Paramètres" > "Système"
7. Ajuster les paramètres selon vos besoins (limites de ressources, priorités, etc.)
8. Vérifier la connexion des nœuds de calcul :
9. Aller dans "Cluster" > "Nœuds"
10. Vérifier que tous les nœuds de calcul sont connectés et actifs

Module de simulation préflop avec ICM

Principes de l'ICM

L'Independent Chip Model (ICM) est un concept fondamental dans les tournois de poker. Il permet de convertir la valeur des jetons en valeur monétaire réelle en fonction de la structure des prix du tournoi.

La formule de base de l'ICM pour un joueur i est :

$EV_i = \sum_{j=1}^{n} P(i \text{ finit à la position } j) \times Prize_j$

Où : - EV_i est l'espérance de gain du joueur i - $P(i \text{ finit à la position } j)$ est la probabilité que le joueur i termine à la position j - $Prize_j$ est le prix attribué à la position j - n est le nombre total de joueurs

L'ICM a un impact significatif sur les décisions optimales en tournoi, particulièrement dans les situations de push/fold en fin de tournoi.

Configuration des simulations préflop

Pour configurer une simulation préflop avec ICM, vous devez spécifier plusieurs paramètres :

Structure du tournoi

```
"tournament_structure": {
    "num_players": 9,
    "player_stacks": [10, 15, 20, 25, 30, 35, 40, 45, 50],  // 
En big blinds
    "blinds": {"sb": 0.5, "bb": 1, "ante": 0.1},
    "prize_structure": [0.5, 0.3, 0.2]  // 50% pour le 1er, 30% 
pour le 2e, 20% pour le 3e
}
```

Ranges des joueurs

```
"player_ranges": {
    "UTG": "22+,A2s+,K7s+,Q9s+,J9s+,T9s,98s,ATo+,KTo+,QTo+,JTo",
    "MP": "22+,A2s+,K5s+,Q8s+,J9s+,T8s+,98s,
87s,ATo+,KTo+,QTo+,JTo",
    "CO": "22+,A2s+,K2s+,Q6s+,J8s+,T8s+,98s,87s,
76s,ATo+,KTo+,QTo+,JTo",
    "BTN": "22+,A2s+,K2s+,Q2s+,J7s+,T7s+,97s+,86s+,76s,
65s,A2o+,K5o+,Q8o+,J8o+,T8o+",
    "SB": "22+,A2s+,K2s+,Q2s+,J2s+,T6s+,96s+,86s+,76s,65s,
54s,A2o+,K2o+,Q6o+,J8o+,T8o+",
    "BB": "22+,A2s+,K2s+,Q2s+,J2s+,T2s+,92s+,82s+,72s+,62s+,
52s+,42s+,32s,A2o+,K2o+,Q2o+,J2o+,T2o+"
}
```

Actions à considérer

```
"actions": {
    "fold": true,
```

```
    "call": true,
    "min_raise": true,
    "custom_raises": [2, 2.5, 3, 4]  // En multiples de la big
blind
}
```

Paramètres de simulation

```
 "num_simulations": 2_000_000_000,  // 2 milliards de simulations
"icm_precision": "high",           // Précision élevée pour
l'ICM
"convergence_threshold": 0.0001    // Seuil de convergence
```

Interprétation des résultats préflop

Les résultats d'une simulation préflop avec ICM sont présentés sous plusieurs formes :

Ranges optimales

Pour chaque position et chaque action, le solver calcule les ranges optimales, présentées sous forme de liste de mains ou de cartes de chaleur.

Exemple de range de push optimale pour le BTN avec 10 BB en ICM :

```
22+,A2s+,K5s+,Q8s+,J8s+,T8s+,98s,87s,76s,A7o+,K9o+,Q9o+,J9o+,T9o
```

Comparaison ICM vs Chip EV

Le solver permet de comparer les stratégies optimales avec et sans ICM, mettant en évidence l'impact de l'ICM sur les décisions.

Exemple de comparaison pour une situation de push/fold :

```
 # Range de push optimale pour le BTN avec ICM
22+,A2s+,K5s+,Q8s+,J8s+,T8s+,98s,87s,76s,A7o+,K9o+,Q9o+,J9o+,T9o

# Range de push optimale pour le BTN sans ICM (Chip EV)
22+,A2s+,K2s+,Q5s+,J7s+,T7s+,97s+,86s+,76s,
65s,A2o+,K7o+,Q8o+,J8o+,T8o+,98o
```

Visualisations

Les résultats sont également présentés sous forme de visualisations interactives :

- **Cartes de chaleur** : Représentation visuelle des ranges avec un code couleur indiquant la fréquence de chaque action.
- **Graphiques d'EV** : Comparaison de l'EV de différentes actions pour chaque main.
- **Tableaux de sensibilité** : Analyse de la sensibilité des stratégies aux variations des paramètres.

Module de simulation postflop

Défis de la simulation postflop

La phase postflop présente des défis spécifiques en raison de l'explosion combinatoire de l'espace d'états :

- **Nombre de boards possibles** : Environ 42,3 millions de boards complets (flop, turn, river)
- **Taille de l'arbre de jeu** : Millions de nœuds pour chaque board
- **Interdépendance des rues** : Les décisions optimales au flop dépendent des stratégies au turn et à la river

Configuration des simulations postflop

Pour configurer une simulation postflop, vous devez spécifier plusieurs paramètres :

Board et structure du pot

```
"board": {
    "flop": ["As", "Kh", "7d"],
    "turn": "2c",
    "river": "9s"
},
"pot_size": 100,  // En BB
"effective_stacks": 200  // En BB
```

Positions et ranges

```
"positions": {
    "flop": "OOP_first",
    "turn": "OOP_first",
    "river": "OOP_first"
},
```

```
"ranges": {
    "OOP": "22+,A2s+,K5s+,Q8s+,J9s+,T8s+,98s,87s,76s,65s,
54s,A7o+,K9o+,QTo+,JTo",
    "IP": "22+,A2s+,K2s+,Q5s+,J7s+,T7s+,97s+,86s+,76s,65s,
54s,A2o+,K5o+,Q8o+,J8o+"
}
```

Actions et tailles de mise

```
"actions": {
    "bet_sizes": [0.33, 0.5, 0.75, 1.0, 1.5, 2.0, "all_in"],
    "raise_sizes": [2.5, 3.0, 4.0, "all_in"]
}
```

Paramètres de simulation

```
"num_simulations": 5_000_000,  // 5 millions de simulations
"streets": ["flop", "turn", "river"],  // Rues à analyser
"abstraction_level": "high"  // Niveau d'abstraction
```

Techniques d'abstraction

Pour gérer la complexité postflop, le solver utilise plusieurs techniques d'abstraction :

Abstraction de cartes

Regroupement des cartes similaires pour réduire le nombre de situations à analyser : - **Abstraction par potentiel** : Regroupement basé sur le potentiel d'amélioration des mains - **Abstraction par équité** : Regroupement basé sur l'équité contre des ranges représentatives - **Abstraction par clusters** : Utilisation d'algorithmes de clustering pour identifier des groupes de mains similaires

Abstraction d'actions

Limitation du nombre d'actions possibles à chaque point de décision : - **Discrétisation des tailles de mise** : Sélection d'un ensemble représentatif de tailles de mise - **Élimination des actions dominées** : Suppression des actions clairement sous-optimales

Interprétation des résultats postflop

Les résultats d'une simulation postflop sont présentés sous plusieurs formes :

Stratégies optimales

Pour chaque point de décision, le solver calcule les stratégies optimales, indiquant la fréquence de chaque action pour chaque main.

Exemple de stratégie pour OOP sur un flop As-Kh-7d :

```
 # Avec les nuts (sets d'As et de K)
- Bet 75% pot (80% du temps)
- Check (20% du temps)

# Avec les mains fortes (deux paires)
- Bet 50% pot (60% du temps)
- Check (40% du temps)

# Avec les draws forts (flush draws)
- Bet 33% pot (40% du temps)
- Check (60% du temps)

# Avec les draws faibles (gutshots)
- Check (100% du temps)

# Avec l'air
- Check (100% du temps)
```

Analyse multi-street

Le solver permet d'analyser les stratégies à travers les différentes rues, montrant comment les décisions au flop influencent les stratégies au turn et à la river.

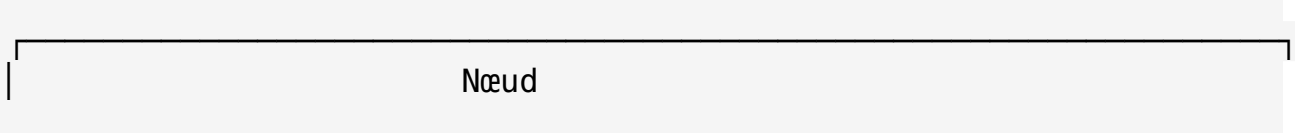
Visualisations

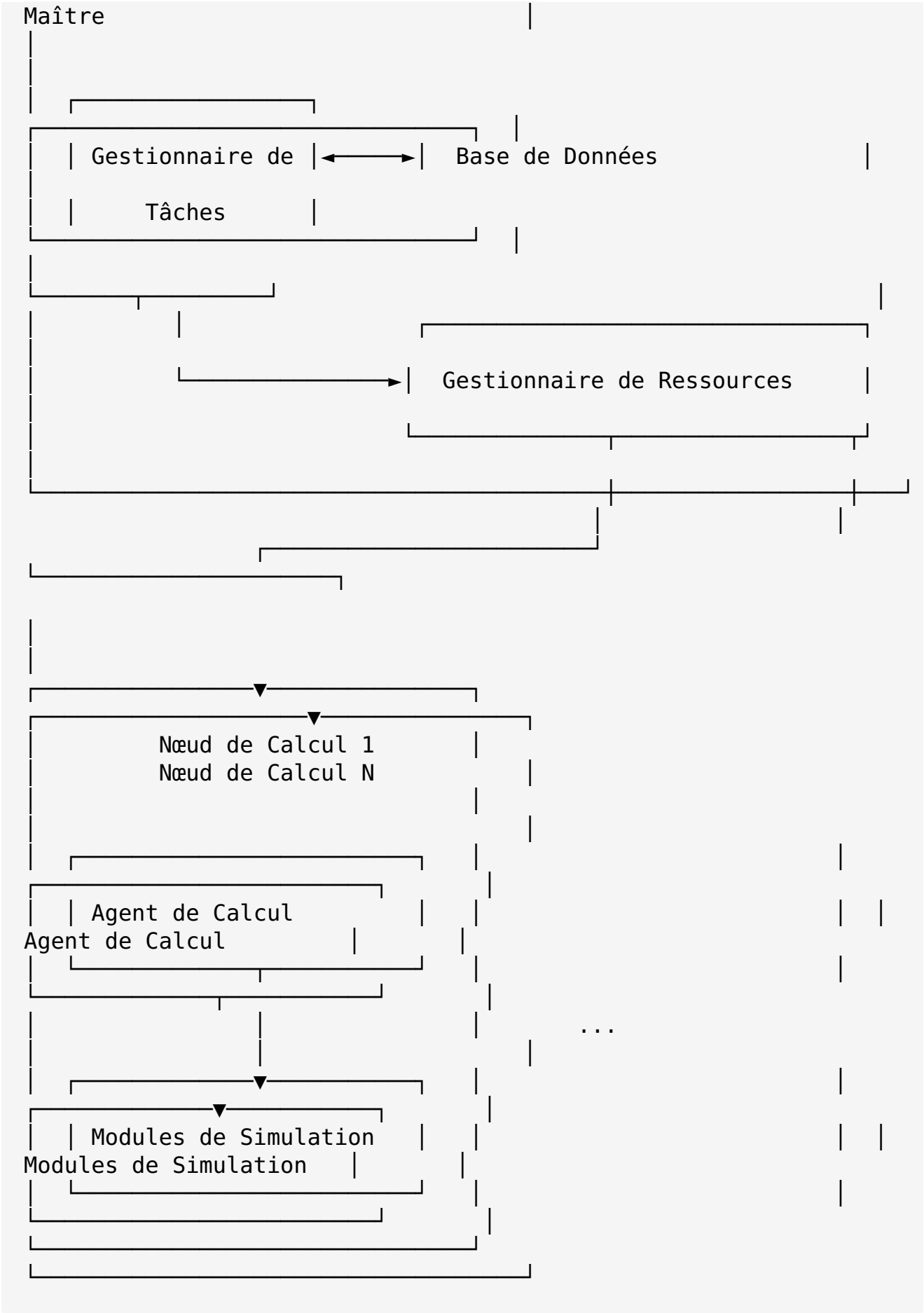
Les résultats sont également présentés sous forme de visualisations interactives : - **Cartes de chaleur** : Représentation visuelle des ranges pour chaque action - **Graphiques d'EV** : Comparaison de l'EV de différentes actions pour chaque main - **Arbres de décision** : Représentation des séquences d'actions optimales

Mode cluster et calcul distribué

Architecture du cluster

Le système de calcul distribué repose sur une architecture maître-esclave :



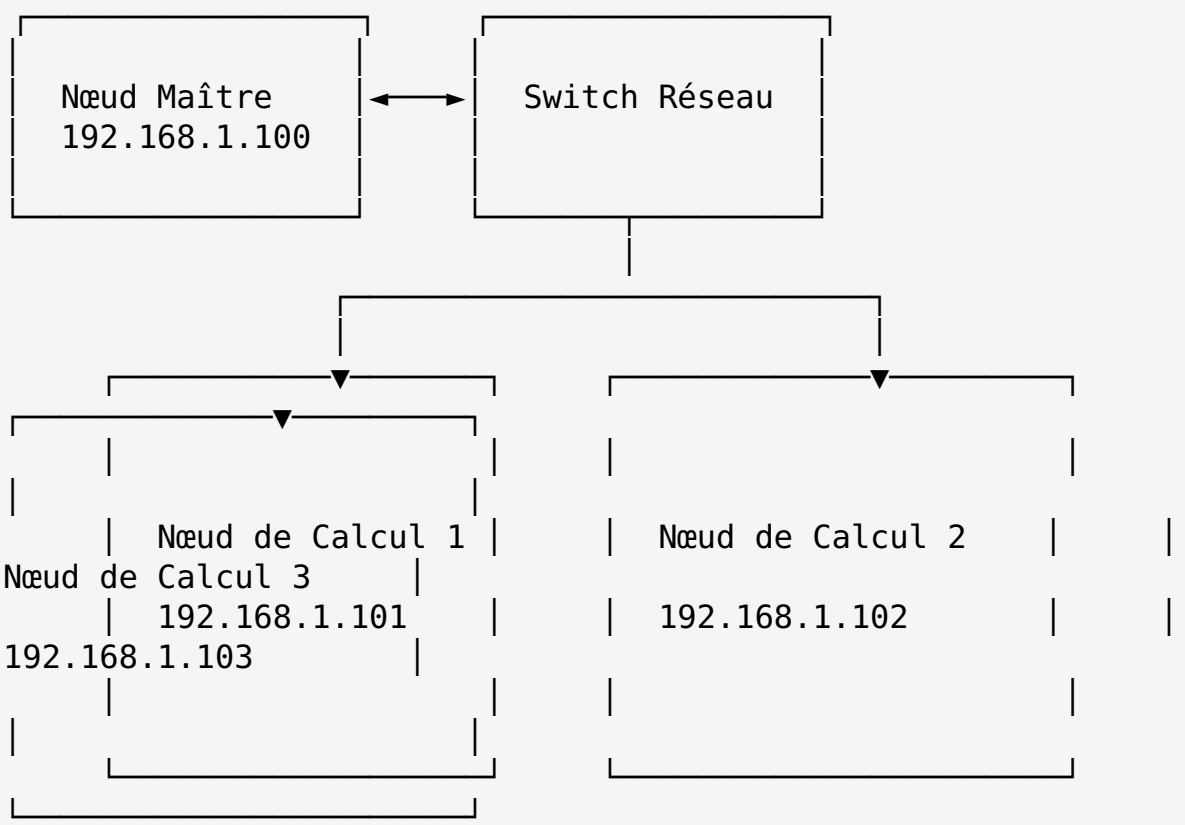

Maître
Gestionnaire de Tâches
Base de Données
Gestionnaire de Ressources
Nœud de Calcul 1
Nœud de Calcul N
Agent de Calcul
Agent de Calcul
...
Modules de Simulation
Modules de Simulation

Configuration d'un cluster local

Pour configurer un cluster local, suivez ces étapes :

1. Installer le nœud maître sur une machine dédiée
2. Installer les nœuds de calcul sur les autres machines
3. Configurer le réseau pour permettre la communication entre les nœuds
4. Enregistrer les nœuds de calcul auprès du nœud maître

Exemple de configuration réseau pour un cluster local :



Configuration d'un cluster distant

Pour les déploiements à plus grande échelle, vous pouvez configurer un cluster distant sur une infrastructure cloud comme AWS, Google Cloud ou Azure.

Exemple de déploiement sur AWS avec Terraform :

```
# Extrait du fichier Terraform pour AWS
resource "aws_instance" "master_node" {
  ami           = "ami-0c55b159cbfafe1f0"  # Ubuntu 20.04
  instance_type = "c5.2xlarge"
  # ... autres configurations
}
```

```
resource "aws_autoscaling_group" "compute_nodes" {
  name                 = "poker-solver-compute-asg"
  min_size             = 3
  max_size             = 10
  desired_capacity     = 3
  # ... autres configurations
}
```

Sécurité et fiabilité

Pour garantir la sécurité et la fiabilité du cluster, plusieurs mécanismes sont mis en place :

Sécurisation des communications

- **Chiffrement TLS/SSL** : Toutes les communications sont chiffrées
- **Authentification mutuelle** : Les nœuds s'authentifient mutuellement
- **Autorisation basée sur les rôles** : Contrôle d'accès granulaire

Tolérance aux pannes

- **Détection des pannes** : Surveillance continue de l'état des nœuds
- **Réplication des tâches** : Exécution redondante des tâches critiques
- **Checkpointing** : Sauvegarde régulière de l'état des calculs
- **Reprise automatique** : Capacité à reprendre les calculs après une panne

Monitoring et administration

Le système offre des outils de monitoring et d'administration pour surveiller l'état du cluster :

- **Tableau de bord** : Interface web pour visualiser l'état du cluster
- **Alertes** : Notification en cas de problème
- **Journalisation centralisée** : Agrégation des logs de tous les nœuds
- **Métriques de performance** : Suivi des performances et de l'utilisation des ressources

Guides d'utilisation

Soumission d'une tâche de simulation préflop

Pour soumettre une tâche de simulation préflop avec ICM :

1. Accéder à l'interface web du nœud maître :

```
http://master-node:8080
```

1. Aller dans "Simulations" > "Nouvelle simulation"
2. Sélectionner "Simulation préflop avec ICM"
3. Configurer les paramètres de la simulation :
4. Structure du tournoi
5. Ranges des joueurs
6. Actions à considérer
7. Paramètres de simulation
8. Cliquer sur "Soumettre"

Vous pouvez également soumettre une tâche via l'API REST :

```python
import requests
import json

# Configuration de la tâche
task_config = {
    "type": "preflop_simulation",
    "icm_enabled": True,
    "num_simulations": 2_000_000_000,
    "tournament_structure": {
        "num_players": 9,
        "player_stacks": [10, 15, 20, 25, 30, 35, 40, 45, 50],
        "blinds": {"sb": 0.5, "bb": 1, "ante": 0.1},
        "prize_structure": [0.5, 0.3, 0.2]
    },
    "player_ranges": {
        "UTG": "22+,A2s+,K7s+,Q9s+,J9s+,T9s,
98s,ATo+,KTo+,QTo+,JTo",
        # ... autres positions
    },
    "actions": {
        "fold": True,
        "call": True,
        "min_raise": True,
        "custom_raises": [2, 2.5, 3, 4]
    },
    "priority": "high"
}

# Soumettre la tâche au nœud maître
response = requests.post(
    "http://master-node:8080/api/jobs",
```

```
    json=task_config,
    headers={"Content-Type": "application/json"}
)

if response.status_code == 200:
    job_id = response.json()["job_id"]
    print(f"Job soumis avec succès. ID: {job_id}")
```

Soumission d'une tâche de simulation postflop

Pour soumettre une tâche de simulation postflop :

1. Accéder à l'interface web du nœud maître
2. Aller dans "Simulations" > "Nouvelle simulation"
3. Sélectionner "Simulation postflop"
4. Configurer les paramètres de la simulation :
5. Board et structure du pot
6. Positions et ranges
7. Actions et tailles de mise
8. Paramètres de simulation
9. Cliquer sur "Soumettre"

Vous pouvez également soumettre une tâche via l'API REST :

```
import requests
import json

# Configuration de la tâche
task_config = {
    "type": "postflop_simulation",
    "num_simulations": 5_000_000,
    "board": {
        "flop": ["As", "Kh", "7d"],
        "turn": "2c",
        "river": "9s"
    },
    "pot_size": 100,
    "effective_stacks": 200,
    "positions": {
        "flop": "OOP_first",
        "turn": "OOP_first",
        "river": "OOP_first"
    },
```

```
    "ranges": {
        "OOP": "22+,A2s+,K5s+,Q8s+,J9s+,T8s+,98s,87s,76s,65s,
54s,A7o+,K9o+,QTo+,JTo",
        "IP": "22+,A2s+,K2s+,Q5s+,J7s+,T7s+,97s+,86s+,76s,65s,
54s,A2o+,K5o+,Q8o+,J8o+"
    },
    "actions": {
        "bet_sizes": [0.33, 0.5, 0.75, 1.0, 1.5, 2.0, "all_in"],
        "raise_sizes": [2.5, 3.0, 4.0, "all_in"]
    },
    "priority": "medium"
}

# Soumettre la tâche au nœud maître
response = requests.post(
    "http://master-node:8080/api/jobs",
    json=task_config,
    headers={"Content-Type": "application/json"}
)

if response.status_code == 200:
    job_id = response.json()["job_id"]
    print(f"Job soumis avec succès. ID: {job_id}")
```

Suivi de l'avancement et récupération des résultats

Pour suivre l'avancement d'une tâche et récupérer les résultats :

1. Accéder à l'interface web du nœud maître
2. Aller dans "Simulations" > "Jobs en cours"
3. Sélectionner le job à suivre
4. L'interface affiche :
5. L'état d'avancement
6. Les ressources utilisées
7. Le temps écoulé et le temps estimé restant
8. Une fois le job terminé, aller dans "Simulations" > "Résultats"
9. Sélectionner le job pour visualiser les résultats

Vous pouvez également suivre l'avancement et récupérer les résultats via l'API REST :

```
import requests
import json
```

```python
import time

job_id = "job_12345"

# Suivre l'avancement
while True:
    status_response = requests.get(f"http://master-node:8080/
api/jobs/{job_id}/status")
    status = status_response.json()["status"]
    progress = status_response.json().get("progress", 0)

    print(f"Statut: {status}, Progression: {progress:.2f}%")

    if status in ["completed", "failed"]:
        break

    time.sleep(10)

# Récupérer les résultats
if status == "completed":
    results_response = requests.get(f"http://master-node:8080/
api/jobs/{job_id}/results")
    results = results_response.json()

    # Sauvegarder les résultats dans un fichier
    with open(f"results_{job_id}.json", "w") as f:
        json.dump(results, f, indent=2)

    print(f"Résultats sauvegardés dans results_{job_id}.json")
```

Administration du cluster

Pour administrer le cluster :

1. Accéder à l'interface web d'administration du nœud maître :

```
http://master-node:8080/admin
```

1. Se connecter avec les identifiants administrateur

2. L'interface d'administration permet de :

3. Gérer les nœuds de calcul (ajout, suppression, mise à jour)
4. Configurer les paramètres du cluster
5. Surveiller les performances
6. Gérer les utilisateurs et les droits d'accès
7. Effectuer des sauvegardes et des restaurations

Ajout d'un nouveau nœud

Pour ajouter un nouveau nœud de calcul :

1. Installer l'agent de calcul sur la nouvelle machine
2. Configurer l'agent pour qu'il se connecte au nœud maître
3. Dans l'interface d'administration, aller dans "Cluster" > "Nœuds" > "Ajouter un nœud"
4. Saisir les informations du nouveau nœud :
5. Nom
6. Adresse IP
7. Capacités (préflop, postflop, GPU)
8. Limites de ressources
9. Cliquer sur "Ajouter"

Mise à l'échelle du cluster

Pour ajuster la taille du cluster en fonction des besoins :

1. Dans l'interface d'administration, aller dans "Cluster" > "Scaling"
2. Configurer les paramètres de scaling :
3. Nombre minimum de nœuds
4. Nombre maximum de nœuds
5. Seuil d'utilisation pour le scaling up
6. Seuil d'utilisation pour le scaling down
7. Activer l'auto-scaling si nécessaire

Exemples pratiques

Exemple 1 : Analyse d'une situation de push/fold en fin de tournoi

Cet exemple montre comment analyser une situation de push/fold en fin de tournoi avec ICM.

Configuration

```
{
    "type": "preflop_simulation",
    "icm_enabled": true,
    "num_simulations": 1_000_000_000,
    "tournament_structure": {
        "num_players": 3,
        "player_stacks": [10, 5, 15],  // BTN, SB, BB en BB
        "blinds": {"sb": 0.5, "bb": 1, "ante": 0},
        "prize_structure": [0.65, 0.35, 0]  // 65% pour le 1er,
35% pour le 2e, 0% pour le 3e
    },
    "focus_position": "BTN",
    "actions": {
        "fold": true,
        "all_in": true
    }
}
```

Résultats

Le solver calcule la range de push optimale pour le BTN :

```
 # Range de push optimale pour le BTN avec ICM
22+,A2s+,K5s+,Q8s+,J8s+,T8s+,98s,87s,76s,A7o+,K9o+,Q9o+,J9o+,T9o
```

Pour comparer, voici la range de push optimale sans ICM (Chip EV) :

```
 # Range de push optimale pour le BTN sans ICM (Chip EV)
22+,A2s+,K2s+,Q5s+,J7s+,T7s+,97s+,86s+,76s,
65s,A2o+,K7o+,Q8o+,J8o+,T8o+,98o
```

On observe que la range de push avec ICM est significativement plus serrée que la range Chip EV, illustrant l'effet de la pression ICM sur les décisions optimales.

Exemple 2 : Analyse d'une situation postflop complexe

Cet exemple montre comment analyser une situation postflop complexe.

Configuration

```
{
    "type": "postflop_simulation",
    "num_simulations": 5_000_000,
```

```
    "board": {
        "flop": ["9s", "8s", "7c"],
        "turn": "6d",
        "river": "Jh"
    },
    "pot_size": 20,
    "effective_stacks": 100,
    "positions": {
        "flop": "OOP_first",
        "turn": "OOP_first",
        "river": "OOP_first"
    },
    "ranges": {
        "OOP": "22+,A2s+,K5s+,Q8s+,J9s+,T8s+,98s,87s,76s,65s,
54s,A7o+,K9o+,QTo+,JTo",
        "IP": "22+,A2s+,K2s+,Q5s+,J7s+,T7s+,97s+,86s+,76s,65s,
54s,A2o+,K5o+,Q8o+,J8o+"
    },
    "actions": {
        "bet_sizes": [0.33, 0.5, 0.75, 1.0, 1.5, 2.0, "all_in"],
        "raise_sizes": [2.5, 3.0, 4.0, "all_in"]
    }
}
```

Résultats

Le solver calcule les stratégies optimales à chaque rue :

```
 # Stratégie au flop
- OOP bet 50% pot avec : sets, straights, deux paires, flush
draws, straight draws
- IP call avec : paires, draws, overcards

# Stratégie au turn
- OOP check avec toute sa range
- IP bet 75% pot avec : straights, sets, deux paires, flush
draws
- OOP call/raise avec : straights, sets, flush draws
- OOP fold avec : paires faibles, busted draws

# Stratégie à la river
- OOP check avec toute sa range
- IP bet 100% pot avec : straights, sets, bluffs sélectionnés
- OOP call avec : straights, sets, deux paires fortes
- OOP fold avec : paires, busted draws
```

Cette analyse multi-street montre comment les stratégies évoluent à travers les différentes rues, avec des ajustements en fonction des cartes qui apparaissent et des actions précédentes.

Dépannage et FAQ

Problèmes courants et solutions

Le nœud maître ne démarre pas

Problème : Le service du nœud maître ne démarre pas ou s'arrête immédiatement après le démarrage.

Solutions : 1. Vérifier les logs :

```
journalctl -u poker_solver_master.service
```

1. Vérifier la configuration :

```
python3 scripts/check_config.py --config config/
master_config.json
```

1. Vérifier les dépendances :

```
python3 scripts/check_dependencies.py
```

Un nœud de calcul ne se connecte pas au nœud maître

Problème : Un nœud de calcul ne parvient pas à se connecter au nœud maître.

Solutions : 1. Vérifier la connectivité réseau :

```
ping master-node
```

1. Vérifier les pare-feu :

```
sudo iptables -L
```

1. Vérifier la configuration du nœud de calcul :

```
python3 scripts/check_config.py --config config/
compute_node.conf
```

1. Vérifier les logs du nœud de calcul :

```
journalctl -u poker_solver_compute.service
```

Les simulations sont trop lentes

Problème : Les simulations prennent beaucoup plus de temps que prévu.

Solutions : 1. Vérifier l'utilisation des ressources :

```
htop
```

1. Vérifier l'équilibrage de charge :

```
http://master-node:8080/admin/monitoring
```

1. Ajuster les paramètres de simulation :
2. Réduire le nombre de simulations
3. Augmenter le niveau d'abstraction
4. Simplifier l'arbre de jeu
5. Ajouter plus de nœuds de calcul au cluster

FAQ

Quelle est la différence entre ICM et Chip EV ?

Réponse : L'ICM (Independent Chip Model) convertit la valeur des jetons en valeur monétaire réelle en fonction de la structure des prix du tournoi, tandis que le Chip EV (Chip Expected Value) considère uniquement la valeur nominale des jetons. En tournoi, l'ICM est généralement plus pertinent car il tient compte de l'impact de chaque décision sur la valeur réelle du stack.

Combien de nœuds de calcul sont nécessaires pour des simulations efficaces ?

Réponse : Le nombre optimal de nœuds dépend de la complexité des simulations et du temps de réponse souhaité. Pour des simulations préflop avec ICM, 3-5 nœuds peuvent suffire pour des résultats en quelques heures. Pour des simulations postflop complexes, 10+ nœuds peuvent être nécessaires pour obtenir des résultats dans un délai raisonnable.

Comment interpréter les fréquences d'action dans les résultats ?

Réponse : Les fréquences d'action indiquent la proportion du temps où chaque action doit être prise avec une main donnée. Par exemple, si le solver indique "Bet 75% pot (80% du temps), Check (20% du temps)" pour une main, cela signifie que vous devriez miser 75% du pot dans 80% des cas et checker dans 20% des cas avec cette main. Ces stratégies mixtes sont essentielles pour être imprévisible et éviter d'être exploité.

Le solver prend-il en compte les tells et les tendances des adversaires ?

Réponse : Non, le solver calcule des stratégies d'équilibre qui sont optimales contre des adversaires parfaits. Pour exploiter les tendances spécifiques des adversaires, vous pouvez utiliser la fonctionnalité de "nodelocking" pour forcer certaines stratégies pour l'adversaire et laisser le solver calculer la contre-stratégie optimale.

Comment simplifier les stratégies du solver pour une application pratique ?

Réponse : Les stratégies du solver peuvent être simplifiées de plusieurs façons : 1. Regrouper les mains similaires 2. Arrondir les fréquences (par exemple, 80% devient 100%, 20% devient 0%) 3. Réduire le nombre de tailles de mise 4. Se concentrer sur les principes généraux plutôt que sur les détails spécifiques

Références et ressources

Documentation API

La documentation complète de l'API REST est disponible à l'adresse suivante :

```
http://master-node:8080/api/docs
```

Les principales endpoints sont :

- `POST /api/jobs` : Soumettre un nouveau job
- `GET /api/jobs/{job_id}/status` : Obtenir l'état d'un job
- `GET /api/jobs/{job_id}/results` : Récupérer les résultats d'un job
- `GET /api/nodes` : Lister les nœuds de calcul
- `POST /api/nodes` : Ajouter un nouveau nœud de calcul
- `GET /api/system/status` : Obtenir l'état du système

Ressources d'apprentissage

Pour approfondir vos connaissances sur les concepts utilisés dans le solver :

- Guide de l'ICM
- Comprendre les solvers GTO
- Techniques d'abstraction pour le poker
- Calcul distribué pour les simulations de poker

Communauté et support

Pour obtenir de l'aide et partager vos expériences :

- Forum de discussion
- Canal Discord
- Base de connaissances
- Support technique

Mises à jour et développement

Pour suivre les mises à jour et contribuer au développement :

- Dépôt GitHub
- Notes de version
- Feuille de route
- Guide de contribution
