+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Mis à jour le 18/06/2025

S'inscrire

Formation OpenMP

3 jours (21 heures)

Présentation

Notre formation OpenMP vous permettra de découvrir et maîtriser cette API puissante dédiée à la programmation parallèle sur architectures à mémoire partagée. Vous apprendrez à exploiter pleinement les processeurs multi-cœurs pour rendre vos applications plus rapides, plus efficaces et prêtes pour le calcul intensif.

Dans cette formation, vous commencerez par les bases du parallélisme : création de threads, gestion du partage de mémoire, et premières directives #pragma pour structurer votre code en régions parallèles simples.

Vous apprendrez ensuite à paralléliser des boucles, à manipuler les variables partagées ou privées, et à agréger les résultats sans erreurs grâce à la directive reduction. L'objectif : écrire du code parallèle sûr et performant.

Nous aborderons également la planification dynamique, la synchronisation fine avec critical, atomic et barrier, ainsi que la création de tâches indépendantes via sections et task.

Comme pour toutes nos formations, celle-ci vous sera présentée avec la toute dernière version de OpenMP.

Objectifs

- Comprendre les principes du parallélisme en mémoire partagée et les cas d'usage d'OpenMP pour accélérer les traitements sur CPU
- Utiliser les directives OpenMP pour créer des régions parallèles et exploiter efficacement les cœurs du processeur
- Paralléliser les boucles de calcul avec parallel for, gérer les variables partagées et privées, et sécuriser les agrégations avec reduction
- Maîtriser la synchronisation des threads avec les directives critical, atomic, barrier et structurer le code en sections indépendantes

- Contrôler la planification du travail entre threads avec schedule, et optimiser les performances selon la charge de travail
- Mettre en œuvre des tâches dynamiques avec task et taskwait pour paralléliser des traitements non uniformes
- Analyser les performances avec omp_get_wtime, identifier les points critiques et éviter les erreurs courantes de concurrence
- Développer un mini-projet de bout en bout en appliquant les concepts appris sur un cas réel à forte intensité de calcul
- Adopter les bonnes pratiques de développement OpenMP pour produire du code clair, efficace et facilement maintenable

Public visé

- Développeur fullstack
- Développeur front-end

Pré-requis

- Bonne maîtrise du langage C ou C++
- Notions de compilation et d'utilisation de la ligne de commande
- Connaissances de base en algorithmique et en manipulation de tableaux ou structures de données

PROGRAMME DE NOTRE FORMATION OpenMP

Introduction au parallélisme et à OpenMP

- Les processeurs modernes comportent plusieurs cœurs : pour profiter pleinement du matériel, il faut exécuter des calculs en parallèle.
- Le parallélisme améliore les performances, surtout sur les traitements lourds (calculs scientifiques, IA, simulations…).
- Deux grands types de parallélisme :
  - Mémoire partagée : tous les threads partagent la même RAM (OpenMP)
  - Mémoire distribuée : chaque processus a sa mémoire (MPI)
- OpenMP (Open Multi-Processing) est une API pour C, C++ et Fortran.
- Permet de paralléliser sans gérer manuellement les threads (vs pthread ou std::thread).
- Basée sur des directives de compilation (#pragma omp), elle est simple à intégrer dans un code existant.
- Compatible avec GCC, Clang, Intel, MSVC…
- Compiler : gcc ou clang avec l'option -fopenmp

Premiers pas avec les threads

- Directive de base
- Utilisation de "omp_get_thread_num()" pour connaître l'identité du thread.
- "omp_get_num_threads()" : nombre de threads utilisés
- "omp_set_num_threads(n)" : fixer le nombre avant la région parallèle

- Objectif : afficher un message depuis chaque thread
- Observation : threads ne s'exécutent pas dans l'ordre

Paralléliser les boucles

- Permet de répartir les itérations d'une boucle entre threads :
- Par défaut, certaines variables sont partagées (globales), d'autres privées (locales à chaque thread).
- Appliquer une transformation à un grand tableau en parallèle
- Mesurer le gain de temps avec "omp_get_wtime()"

Réduction et agrégation des résultats

- Si plusieurs threads modifient une variable partagée (ex : somme), ça cause une condition de course.
- Directive OpenMP qui permet d'agréger les résultats sans collision
- Comparer version séquentielle, version parallèle naïve (bug), version avec reduction
- Analyser les erreurs liées aux variables partagées

Contrôle des tâches et synchronisation

- #pragma omp barrier : attend que tous les threads arrivent
- #pragma omp critical : section protégée, un seul thread à la fois
- #pragma omp atomic : opération atomique plus rapide mais limitée
- Simuler un traitement en 3 étapes (lecture, calcul, sauvegarde) avec sections

Gestion des performances et planification

- Contrôle de la distribution du travail :
  - Static : parts fixes
  - Dynamic : planification à la volée
  - Guided : dynamique mais décroissante
- Tester différentes stratégies de planification sur une boucle où les temps de traitement varient

Paralléliser un projet réel

- Version séquentielle
- Analyse des dépendances
- Parallélisation progressive avec OpenMP
- Optimisation : éviter les accès mémoire coûteux
- Paralléliser un programme au choix (filtrage image, tri, statistiques…)
  - Identifier les boucles critiques
  - Ajouter les directives
  - Mesurer les performances
  - Éviter les conditions de course

Bonnes pratiques et perspectives

- Variables mal déclarées (shared vs private)
- Accès concurrent sans critical ou reduction
- Trop de threads = surcharge CPU
- Toujours mesurer les performances
- Paralléliser progressivement
- Favoriser les traitements indépendants
- Comparaison avec MPI, CUDA, OpenACC
- OpenMP + SIMD
- Profilage avec gprof, perf, Intel VTune

Sociétés concernées

Cette formation s'adresse à la fois aux particuliers ainsi qu'aux entreprises, petites ou grandes, souhaitant former ses équipes à une nouvelle technologie informatique avancée ou bien à acquérir des connaissances métiers spécifiques ou des méthodes modernes.

Positionnement à l'entrée en formation

Le positionnement à l'entrée en formation respecte les critères qualité Qualiopi. Dès son inscription définitive, l'apprenant reçoit un questionnaire d'auto-évaluation nous permettant d'apprécier son niveau estimé sur différents types de technologies, ses attentes et objectifs personnels quant à la formation à venir, dans les limites imposées par le format sélectionné. Ce questionnaire nous permet également d'anticiper certaines difficultés de connexion ou de sécurité interne en entreprise (intraentreprise ou classe virtuelle) qui pourraient être problématiques pour le suivi et le bon déroulement de la session de formation.

Méthodes pédagogiques

Stage Pratique : 60% Pratique, 40% Théorie. Support de la formation distribué au format numérique à tous les participants.

Organisation

Le cours alterne les apports théoriques du formateur soutenus par des exemples et des séances de réflexions, et de travail en groupe.

Validation

À la fin de la session, un questionnaire à choix multiples permet de vérifier l'acquisition correcte des compétences.

Sanction

Une attestation sera remise à chaque stagiaire qui aura suivi la totalité de la formation.