CMake versus CMT

Grigory Rybkin

Laboratoire de l'Accélérateur Linéaire
Orsay

réunion de travail, le 9 novembre 2010

Plan

Plan

Plan

Plan

Réunion de travail CMT en ATLAS

- discutait les mesures des performances CMake et CMT faites dans la présentation de Pere Mato, en particulier, montrait que pour le projet GAUDI CMake est plus rapide de
  1. $\simeq$ 2 fois avec `make` simple
  2. $\simeq$ 4 fois avec `make -j8` dit parallèle
- il fallait comprendre l'origine de telles différences
- pour les détails voir CMT en ATLAS, LAL, le 7 octobre 2010

Optimisation de Code

Configuration CMake — dans les tests qu'on est en train de discuter — construisait code *sans optimisation* tandis que celle du CMT — *avec optimisation* `-O2`, coûteuse en temps de compilation. Cela explique la différence 1

Table: Les mesures du temps réel en secondes sur une machine de 16 cœurs quand j'ai appliqué l'optimisation -02 pour le projet GAUDI

CMake +make+install	CMT/v1r22 cmt br make	CMake +make+install -j8	CMT/v1r22 cmt br make -j8
1080	1110	240	400

CMake toujours utilise mieux `make` parallèle

Mesures Plus Détailées

Table: Les mesures du temps réel en secondes sur une machine de 16 cœurs pour le projet GAUDI avec *make* simple

	CMake	CMT
generation	10	40
g++	960	1010
install	10	
total	1060	1150

Plus de Parallélisme avec CMT

En même temps construire des paquets indépendants dans

[tbroadcast] *tâches légères/fils d'exécution (threads en anglais)*

[Makefile au niveau projet] *processus*

différents

Table: Les mesures du temps réel en secondes sur une machine de 16 cœurs pour le projet GAUDI

tbroadcast make	niveau projet make	tbroadcast make -j8	niveau projet make -j8
990	1120	360	350

Parallélisme avec CMake

En même temps construire des cibles indépendantes dans *processus* différents

Table: Les mesures du temps réel en secondes sur une machine de 16 cœurs pour le projet GAUDI quand j'ai ajouté à la configuration CMake les dépendances entre les paquets équivalentes à celles du CMT

CMake avec dep +make+install	niveau projet make	CMake avec dep +make+install -j8	niveau projet make -j8
1080	1120	330	350

Cela éxplique la différence 2

Nouvelles de Pere Mato

Pere a

- réussi à enchaîner les projets et les cibles d'exportation et d'importation entre eux. C'est important pour simplifier les dépendances de bibliothèques
- migré les tests GAUDI vers CTest
- exploré CPack pour générer tarballes ou RPMs

Conclusion

- facteur $\simeq 2$ de la différence des performances CMake et CMT éxpliqué par la différence des optimisations de code
- CMake peut gagner en performance grâce à l'construction en parallèle, dependant de
  - la structure du projet et des paquets
  - la disponibilité des ressources informatiques
- tbroadcast et Makefile au niveau projet donnent plus de parallélisme avec CMT
- à explorer comment introduire encore plus de parallélisme