* Agent.cpp or Agent.hpp

각 Agent들의 기본적인 부모클래스

in\_policy?

out\_policy?

* Controller.cpp

컨트롤 알고리즘 실행

Controller::~Controller()

종료 signal 수신 때 까지 알고리즘을 계속 수행

**void** Controller::create\_agents(**void**)

모든 Agent의 init() 함수 호출, agent는 signal, controls을 push 할 수있음

**void** Controller::init\_agents(**void**)

Agent의 초기화?(tree에 대한 level로 초기화를 함, 레벨마다 하는 일이 다른가?)

**void** run(**void**);

제어 알고리즘의 한 단계를 실행합니다.  
한 단계는 리소스 관리자로부터 정책 정보를 수신하고, 노드가 부모인 다른 모든 컨트롤러에 보내고, 하드웨어 원격 측정을 읽는 것으로 구성됩니다.

알아볼 것

geopm\_signal\_handler\_check();

m\_application\_io이게뭐지?

**void** Controller::generate(**void**)

제어 알고리즘을 별도의 스레드로 실행합니다.  
종료 신호가 수신될 때까지 제어 알고리즘을 계속 실행하는 POSIX 스레드를 작성합니다.  
이는 제어중인 애플리케이션에 의해 호출된다.이 발사 방법을 사용하면 MPI\_THREAD\_MUTIPLE에 대해 MPI\_Init\_thread()를 사용하여 지원 MPI 구현을 활성화해야 한다.

알아볼 것

M\_agent 이게뭐지?

**void** walk\_down(**void**);

하드웨어 원격 측정 및 응용 프로그램 데이터를 읽고 결합된 데이터를 트리 위로 리소스 관리자로 보냅니다.  
트리의 각 수준에서 에이전트는 샘플을 수정하고 집계하여 부모에게 보낼 수 있습니다

**void** walk\_up(**void**);

보고서 파일을 작성하고 추적을 완료합니다.

**void** step(**void**);

트리의 루트에 있는 리소스 관리자에서 모든 노드의 컨트롤러로 정책 정보를 전파합니다.  
트리의 각 레벨에서 에이전트는 정책을 수정하여 자식에게 보낼 수 있다.