# Sun™ Grid Engine, Enterprise 버전 5.3 관리 기본사항

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 U.S.A.
650-960-1300

부품 번호 : 816-7482-10
2002년 7월, 개정판 A

이 문서에 대한 의견은 다음 주소로 보내십시오: docfeedback@sun.com

# 목차

# Sun Grid Engine, Enterprise 버전 5.3 관리 기본사항

이 문서는 Sun™ Grid Engine, Enterprise 버전 5.3 시스템의 관리를 돕기 위한 빠른 참조 핸드북으로 제공됩니다. 그러한 관리에 관련된 일반 업무에 대한 지침 외에, 이 문서에는 일반적인 그리드 컴퓨팅 및 특히 Sun Grid Engine, Enterprise 버전 5.3 제품의 간략한 설명이 들어 있습니다. 이 문서는 여러 가지 미세 조정 및 문제 해결 정보로 완결됩니다.

이 핸드북을 Sun Grid Engine, Enterprise 버전 5.3 개념 및 절차의 자세한 설명이 있는 *Sun Grid Engine, Enterprise 버전 5.3관리 및 사용 설명서*에 대한 (대체가 아닌) 부록으로 사용하십시오.

다음 업무를 수행하기 위한 지침이 이 문서에 들어 있습니다.

■ 29 페이지의 “문제점 진단”

---

**참고 –** 이 문서에 있는 많은 내용은 원래 Sun Grid Engine 프로젝트 웹사이트의 "How-To" 섹션에 있었습니다. 자주 갱신되는 이 웹사이트는 Sun Grid Engine, Enterprise 버전 5.3 시스템 관리자에게 특별한 가치가 있으며 때때로 참조할 가치가 있습니다. 웹사이트에 대한 URL은 `http://gridengine.sunsource.net/project/gridengine/howto/howto.html`입니다.

---

# 그리드 컴퓨팅과 Sun Grid Engine, Enterprise 버전 5.3 소프트웨어

*그리드* 컴퓨팅 관리가 처음인 사용자라 해도 압도적으로 많은 전문 컴퓨터 관리자에 속합니다. 권능을 부여하는 소프트웨어 기술이 10년이 되지 않기 때문에 아직은 지구상의 극소수의 사람만이 진정한 그리드 컴퓨팅 환경의 숙련된 관리자라고 주장할 수 있습니다. 그리드 컴퓨팅 분야의 개척자 중 한 사람인 이안 포스터 박사에 따르면 그리드 컴퓨팅과 관련된 개념은 1995년까지 단지 "개척" 중이었습니다. 그러므로 그리드 기술의 최초 채택자조차도 10년 미만의 경험을 갖고 있습니다.

그러나 그리드 컴퓨팅에 대한 경험이 적은 것에 상관없이 사용자는 여러 가지 종류의 그리드를 통해 그리드 사용에 관한 방대한 경험을 갖고 있습니다. 전기 스위치를 "켤" 때마다 그리드의 자원을 사용하고 있습니다. 대부분의 산업화된 국가에서 전기는 그리드라고 부르는 공통적인 자원 풀에 제품을 공급하는 수많은 독립 전력 생산자로 구성되는 "전력망(power grid)"을 통해 가정과 사업체에 공급됩니다.

자원의 풀로 만드는 것은 소비자와 생산자 모두에게 이익입니다. 전력 생산자는 지리적으로 인접한 지역의 소비자가 발전소가 생산할 수 있는 만큼의 전기가 필요하지 않은 경우에도 전체 용량으로 발전소를 가동할 수 있게 되어 이익을 얻습니다. 예를 들어 도시에서 멀리 떨어진 전원 지역의 소비자는 대개 일반적인 발전소가 생산할 수 있는 만큼의 많은 전력이 필요하지 않을 것입니다. 그러나 전력망 배열로 발전소가 "과잉" 용량을 그리드에 판매할 수 있게 하여 자체 지역 발전소로는 지역 수요를 충족시킬 수 없는 먼 지역의 소비자에게 전력을 서비스할 수 있습니다.

해당 수요가 어디에 위치할 지에 상관없이 과소 이용된 자원을 생산에 투입하고 수요를 충족시키기 위해 해당 자원을 모으는 능력은 전력망의 정확한 특성입니다. 또한 "그리드" 이름을 채택하여 그리드 컴퓨팅 개념의 개척자들은 적당한 유사성을 만들어 냈습니다. 그리드 컴퓨팅의 기본 목적은 수백(심지어 수천) 대의 유휴 및 거의 유휴인 컴퓨터의 과소 이용된 자원을 풀로 만들고 결합된 최종 컴퓨팅 능력을 연산 집약적인 소비자의 수요를 충족시킬 수 있게 하는 것입니다.

# 분산 자원 관리의 역할

유사성이 다소 깨지는 곳은 세부적인 전달입니다. 전력망은 그리드의 모든 생산자 구성원과 최후로 전기 소비자의 가정, 공장, 농장 및 사무실을 말 그대로 함께 배선하여 자원을 풀로 만들기 위해 관리합니다. 일단 연결되고 전력망에 공급되면 전기는 수요에 따라서 흐릅니다.전등 스위치를 "켜면" 전기가 어디엔가 있는 발전소로부터 전력망을 통해 전구로 흐릅니다.그리드 컴퓨팅 환경의 컴퓨팅 자원도 네트워크 기술에 의한 것이지만 함께 "배선"된다고 말할 수 있습니다(1차적인 예로 인터넷 뿐 아니라 인트라넷까지). 그러나 단순히 네트워크 컴퓨터의 스위치를 켜고 컴퓨팅 그리드의 자원을 사용(또는 자원을 기여)할 것을 기대할 수는 없습니다.

컴퓨팅 그리드에 대한 액세스는 특별한 *미들웨어* 소프트웨어를 설치한 후에만 가능합니다. 미들웨어는 Sun의 Solarisô 운영 환경 또는 Linux 운영 체제 같은 컴퓨터 운영 체제와 3차원 그래픽 렌더링 프로그램 같은 응용 프로그램 소프트웨어 사이에 놓이는 소프트웨어 계층입니다. Sun Grid Engine 소프트웨어는 컴퓨팅 그리드에 참여할 수 있게 하는 미들웨어입니다.

초기의 "개방 소스" 그리드 컴퓨팅 프로젝트의 실재하는 결과인 Sun Grid Engine은 사용자에게 복잡한 작업에 대해 훨씬 더 많은 연산 능력을 제공하도록 복수 사용자와 복수 연산 자원(보통 원격으로 위치하는) 사이에서 조정하는 *분산 자원 관리* 소프트웨어입니다. Sun Grid Engine 소프트웨어를 통해 그리드에 결합된 유휴 컴퓨팅 자원이 종종 먼 사용자에게 사용 가능하게 되므로 사용자와 자원 모두의 생산성을 증가시킵니다. 예를 들어 보통 실제 용량의 1/4로 실행한 컴퓨팅 자원은 그리드에의 참여를 통해 거의 100%로 실행 중인 것으로 표시되었습니다.

연산 장치는 "분산 자원 관리" 용어의 "분산 자원" 부분에 해당합니다. Sun Grid Engine 소프트웨어는 "관리" 용어의 필수적인 부분을 설명합니다. 그러한 관리가 없으면 혼란이 발생합니다. 그리드 관리자로서 Sun Grid Engine 소프트웨어는 사용자가 제출하는 작업을 승인하고, 자원 관리 *정책*을 바탕으로 그리드의 적당한 시스템에서 실행하도록 작업을 예약하는데, 정책은 조직의 기술 및 경영진에 의해 설정됩니다.

두 그리드는 비슷하지 않습니다. 즉, 한 크기가 모든 상황에 맞지 않습니다. 세 개의 핵심 그리드 클래스가 있는데, 이것은 단일 시스템부터 수 천개의 프로세서를 이용하는 수퍼 컴퓨터급의 compute farm까지의 범위를 갖습니다.

- 기본 Sun Grid Engine 5.3 소프트웨어에 의해 활성화되는 *클러스터 그리드*는 가장 단순하며 단일 프로젝트나 부서의 사용자에게 단일 액세스 지점을 제공하기 위해 함께 작동하는 컴퓨터 *호스트*로 구성됩니다.

  Sun Grid Engine, Enterprise 버전 5.3 소프트웨어는 이 기본 모델을 확장하여 다른 두 개의 보다 복잡하고 강력한 그리드 클래스를 작성합니다.

- *캠퍼스 그리드*는 한 조직의 복수 프로젝트 또는 부서가 컴퓨팅 자원을 공유할 수 있게 합니다. 조직은 캠퍼스 그리드를 사용하여 주기적인 비즈니스 프로세스부터 렌더링, 데이터 마이닝(data mining) 등까지 광범위한 작업을 처리할 수 있습니다.

- *글로벌 그리드*는 아주 큰 가상 시스템을 작성하기 위해 조직 경계를 넘는 캠퍼스 그리드의 모음입니다. 사용자들은 자체 조직 내에서 사용할 수 있는 자원을 훨씬 초과하는 연산 능력에 액세스할 수 있습니다.

Sun Grid Engine, Enterprise 버전 5.3 소프트웨어는 캠퍼스 그리드에 필요한 능력과 유연성을 제공합니다. 이 제품은 캠퍼스의 모든 기존 Sun Grid Engine 클러스터 그리드를 통합하여 캠퍼스 그리드를 작성하기 위한 유연한 전이를 수월하게 하므로 기본 Sun Grid Engine 소프트웨어에 의해 활성화되는 기존 클러스터 그리드에 매우 유용합니다. 제품은 또한 처음으로 그리드 컴퓨팅 모델로의 이동을 만드는 엔터프라이즈 캠퍼스를 위한 좋은 출발점입니다.

# 정책, 호스트 및 데몬

Sun Grid Engine, Enterprise 버전 5.3 소프트웨어는 시스템 관리자가 유지보수하는 엔터프라이즈 자원 정책을 기반으로 연산 능력의 전달을 지휘합니다. Sun Grid Engine, Enterprise 버전 5.3 시스템은 이들 정책을 사용하여 캠퍼스 그리드 내에서 사용할 수 있는 연산 자원을 시험하고 이들 자원을 수집한 후 캠퍼스 그리드 전체에서 사용을 최적화하는 방법으로 자동으로 자원을 할당하고 전달합니다.

캠퍼스 그리드 안에서 협력이 가능하게 하려면 그리드를 사용하는 프로젝트 소유자가 정책을 협상하고 고유한 프로젝트 요구사항에 대한 수동 대체를 위해 정책에서 유연성을 갖고 정책이 자동으로 모니터 및 강제 실행되도록 합니다.

관리자는 사이트에 적합한 모든 사항에 따라서 사용자 정의되는 고급 이용 정책을 정의할 수 있습니다. *Sun Grid Engine, Enterprise 버전 5.3 관리 및 사용 설명서*에서 상세히 설명되는 4 가지 정책이 사용 가능합니다.

- 기능적
- 공유 기반
- 기한
- 대체

Sun Grid Engine, Enterprise 버전 5.3 정책 관리는 사용자의 목표를 달성하기 위해 자동으로 클러스터의 공유 자원 사용을 제어합니다. 높은 우선순위 작업은 우선적으로 작업 지정되며 다른 더 낮은 우선순위의 작업과 경쟁할 때 더 큰 CPU 자격 권리를 받습니다. Sun Grid Engine, Enterprise 버전 5.3 소프트웨어는 모든 작업의 진행을 모니터하고 대응하여 및 정책에 정의된 목적에 따라서 작업의 상대 우선순위를 조정합니다.

사이트의 사용 규칙이 느슨하든지 아니면 엄격하든지 간에 Sun Grid Engine, Enterprise 버전 5.3 정책 모듈이 규칙을 수용합니다. 이 정책 기반 자원 할당은 주, 월 또는 분기 같은 누적 기간 동안 각 사용자, 팀, 부서 및 모든 프로젝트에 시스템 자원의 할당된 지분을 부여합니다.

4 가지 유형의 호스트가 Sun Grid Engine, Enterprise 버전 5.3 시스템에 기본적입니다.

- 마스터
- 실행
- 관리
- 제출

또한 관리자가 *섀도우 마스터 호스트*를 작성하는 것이 바람직합니다. 마스터 호스트가 하나만 있는 반면, 더 큰 가용성을 제공하기 위해 클러스터의 다른 시스템이 섀도우 마스터 호스트로 지정될 수 있습니다. 섀도우 마스터 호스트는 지속적으로 마스터 호스트를 모니터하고, 마스터 호스트가 실패하는 경우 자동으로 및 투명하게 제어를 가정합니다. 이미 클러스터에 있는 작업은 마스터 호스트 실패에 의해 영향을 받지 않습니다.

각 호스트 유형은 *Sun Grid Engine, Enterprise Edition 5.3 관리 및 사용 설명서*에 자세히 설명되어 있습니다.

4가지 데몬이 Sun Grid Engine, Enterprise 버전 5.3 기능성을 제공합니다.

- `sge_qmaster`는 마스터 데몬입니다.
- `sge_schedd`는 스케줄러 데몬입니다.
- `sge_execd`는 실행 데몬입니다.
- `sge_commd`는 통신 데몬입니다.

이러한 각 데몬은 *Sun Grid Engine, Enterprise Edition 5.3 관리 및 사용 설명서*에 자세히 설명되어 있습니다.

이 문서의 나머지는 가장 일반적인 Sun Grid Engine, Enterprise 버전 5.3 관리 업무를 수행하는 방법에 대한 지침을 제공합니다. 추가 백그라운드 정보 및 여기에 제공되지 않는 업무를 수행하는 방법에 대한 지침은 *Sun Grid Engine, Enterprise 버전 5.3 관리 및 사용 설명서*, *Sun Grid Engine 5.3 및 Sun Grid Engine, Enterprise 버전 5.3 참조 설명서* 또는 `man` 페이지를 참조하십시오.

# 공통 관리 업무에 대한 명령줄 지침

**참고 –** 다음의 지침 그룹은 명령줄에서의 명령 사용만을 기반으로 합니다. 모든 명령은 Sun Grid Engine, Enterprise 버전 5.3 그래픽 사용자 인터페이스인 `QMON`에도 대응하는 기능성을 갖습니다. `QMON`을 사용하여 다음 업무를 수행하는 방법에 대한 지침은 *Sun Grid Engine, Enterprise Edition 5.3 관리 및 사용 설명서*를 참조하십시오.

## ▼ 호스트에서 관리 특권을 추가 또는 제거하는 방법

● **관리 특권을 추가하려면 다음 명령을 입력하십시오.**

```
qconf -ah
```

● **관리 특권을 제거하려면 다음 명령을 입력하십시오.**

```
qconf -dh
```

## ▼ 실행 호스트를 추가하는 방법

1. **다음 명령을 입력하여 새 호스트를 관리 호스트로 만듭니다.**

```
qconf -ah
```

2. **이 새 호스트의 루트로서 $SGE_ROOT에서 다음 스크립트를 실행합니다.**

```
install_execd
```

## ▼ 실행 호스트를 제거하는 방법

● **다음 명령을 입력하여 이 호스트와 연관된 대기열을 삭제합니다.**

```
qconf -dq
```

● **다음 명령을 입력하여 호스트를 삭제합니다.**

```
qconf -de
```

## ▼ 제출 호스트를 추가 또는 제거하는 방법

● **호스트를 제출 호스트로 지정하려면 호스트의 명령줄에서 다음 명령을 입력하십시오.**

```
qconf -as
```

- **제출 호스트로서의 호스트 지정을 제거하려면 호스트의 명령줄에서 다음 명령을 입력하십시오.**

```
qconf -ds
```

## ▼ 현재 호스트의 이름을 표시하는 방법

- **관리 호스트의 이름을 표시하려면 다음 명령을 입력하십시오.**

```
qconf -sh
```

- **제출 호스트의 이름을 표시하려면 다음 명령을 입력하십시오.**

```
qconf -ss
```

- **실행 호스트의 이름을 표시하려면 다음 명령을 입력하십시오.**

```
qconf -sel
```

## ▼ 대기열 관리 방법

- **대기열을 추가하려면 다음 명령을 입력하십시오.**

```
qconf -aq
```

- ***파일에서* 대기열을 추가하려면 다음 명령을 입력하십시오.**

```
qconf -Aq
```

- **대기열을 삭제하려면 다음 명령을 입력하십시오.**

```
qconf -dq
```

● **대기열을 수정하려면 다음 명령을 입력하십시오.**

```
qconf -mq
```

● **둘 이상의 대기열의 단일 속성을 변경하려면 다음 명령을 입력하십시오.**

```
qconf -mqattr
```

## ▼ 마스터 호스트 변경 방법

1. **다음 명령을 입력하여 현재 마스터 호스트의 마스터 및 스케줄러 데몬을 중단합니다.**

```
qconf -ks -km
```

2. **다음 기준에 따라서** `$SGE_ROOT/default/common/act_qmaster` **파일을 편집합니다.**

   a. `act_qmaster` **파일에서 현재 호스트 이름을 새 마스터 호스트의 이름으로 대체합니다.**

   이 이름은 `gethostname` 유틸리티에 의해 반환되는 이름과 동일해야 합니다.해당 이름을 얻으려면 새 마스터 호스트에서 다음 명령을 입력하십시오.

```
$SGE_ROOT/utilbin/$ARCH/gethostname
```

   b. `act_qmaster` **파일의 이전 이름을** `gethostname` **유틸리티에 의해 반환되는 이름으로 대체합니다.**

3. **새 마스터 호스트에서 다음 스크립트를 실행합니다.**

```
$SGE_ROOT/default/common/sge5
```

이 명령은 새 마스터 호스트에서 `sge_qmaster` 및 `sge_schedd` 를 시작합니다.

## ▼ 섀도우 마스터 설정 방법

1. **다음 기준에 따라서 `shadow_masters` 파일을 작성합니다.**
   - `$SGE_ROOT/default/common`에 파일을 작성합니다.
   - 1차 마스터 호스트의 이름을 첫 행으로 두고 마스터 책임을 가정하도록 선택된 다른 호스트를 사용자가 원하는 순서로 나열합니다. 예를 들어,

```
system% cat shadow_masters
host1
host2
host3
```

   위의 예에서 `host1`이 1차 마스터 호스트입니다. `host1`이 실패하는 경우 `host2`가 대략 10분 후에 마스터 서버로서 인계합니다. 나아가 `host2`가 실패하는 경우 `host3`이 인계합니다.

2. **올바른 사용 권한을 검증합니다.**

   모든 마스터 섀도우 호스트는 `qmaster` 스풀 디렉토리에 대한 읽기/쓰기 권한이 있어야 합니다.

3. **`sge5` 시작 스크립트를 사용하여 섀도우 데몬을 시작합니다. 각 호스트의 루트로서 다음 명령을 입력하십시오.**

```
$SGE_ROOT/default/common/sge5 -shadowd
```

이들 단계를 완료한 후 Sun Grid Engine, Enterprise 버전 5.3 클러스터의 마스터 섀도윙이 활성화됩니다.

# 관리 업무를 위한 스크립트 및 파일 사용

---

**참고 –** `QMON` 그래픽 사용자 인터페이스를 사용하여 모든 Sun Grid Engine, Enterprise 버전 5.3 관리 업무를 수행할 수 있으며, 이를 사용하면 추가로 시스템의 모든 능력에 대해 배울 수 있습니다. 그러나 쉘 프롬프트에서 발행되고 쉘 스트립트 안에서 호출되는 명령을 통해 Sun Grid Engine, Enterprise 버전 5.3 그리드를 관리할 수도 있습니다. 많은 숙련된 관리자는 이것이 설정을 변경하는 더욱 유연하고 빠르며 보다 강력한 방법임을 알 수 있습니다. 다음 세 세트의 지침이 그런 방법을 다룹니다.

---

## ▼ 파일을 사용하여 오브젝트를 추가 또는 수정하는 방법

● **사용자가 파일에 작성하는 스펙에 따라서 오브젝트를 추가 또는 수정하려면 `qconf` 명령을 사용하십시오.**

명령은 다음 구문을 갖습니다. `qconf -{A,M}`<*오브젝트*> <*파일이름*>

위의 구문에서 `-A`는 추가를 의미하고 `-M`은 수정을 의미합니다. <*오브젝트*> 변수는 다음 중 하나일 수 있습니다.

- `c` - 복합
- `ckpt` - 체크포인트 환경
- `e` - 실행 호스트
- `p` - 병렬 환경
- `q` - 대기열
- `u` - 사용자 세트

이 옵션을 `qconf` 명령의 *show* 옵션과 함께 조합하여 사용하여(`qconf -s`<*오브젝트*>) 기존 오브젝트를 취하고 수정한 후 기존 오브젝트를 갱신하거나 새 오브젝트를 작성할 수 있습니다.

### 예

다음은 기존 체크포인트 환경의 *이주 명령*을 수정하는 쉘 스크립트의 예입니다.

```
#!/bin/sh
# ckptmod.sh: modify the migration command
# of a checkpointing environment
# Usage: ckptmod.sh <checkpoint-env-name> <full-path-to-
command>
TMPFILE=/tmp/ckptmod.$$

CKPT=$1
MIGMETHOD=$2

qconf -sckpt $CKPT | grep -v '^migr_command' > $TMPFILE
echo "migr_command $MIGMETHOD" >> $TMPFILE
qconf -Mckpt $TMPFILE
rm $TMPFILE
```

## ▼ 파일을 사용하여 대기열, 호스트 및 환경을 수정하는 방법

두 가지 방법 중 하나로 qconf 명령을 사용하고 명령에 대한 인수에 따라서 다른 명령과 결합하여 명령줄로부터 개별 대기열, 호스트 및 병렬 및 체크포인팅 환경 모두를 수정할 수 있습니다.다음 지침은 둘 다 포함합니다.

- ***이미 파일을 준비한 경우* qconf 명령과 적당한 옵션을 입력하십시오.**

  명령은 다음 구문을 갖습니다. 대문자와 소문자 사이의 차이가 크므로 문자 옵션의 대소문자에 특히 주의하십시오.

  `qconf -M{q,e,p,ckpt}` <*파일이름*>

  옵션은 다음 의미를 갖습니다.

  - `-M` - 기존 파일인 <*파일이름*>에서 수정합니다.
  - `q` - 대기열
  - `e` - 실행 호스트
  - `p` - 병렬 환경
  - `ckpt` - 체크포인트 환경

- ***아직 파일을 준비하지 않은 경우* qconf 명령과 적당한 옵션을 입력하십시오.**

  명령은 다음 구문을 갖습니다. 대문자와 소문자 사이의 차이가 크므로 문자 옵션의 대소문자에 특히 주의하십시오.

  `qconf -m{q,e,p,ckpt}` <*파일이름*>

  옵션은 다음 의미를 갖습니다.

  - `-m` - 수정 파일을 작성할 텍스트 편집 프로그램을 열어서 수정합니다.
  - `q` - 대기열
  - `e` - 실행 호스트
  - `p` - 병렬 환경
  - `ckpt` - 체크포인트 환경

### 논의

명령의 차이, 즉 첫 번째에 있는 대문자 `M`과 두 번째에 있는 소문자 `m`은 명령의 결과를 제어합니다. `-M` 및 `-m` 옵션 모두가 *수정*을 의미하지만 대문자 `-M`은 *기존* 파일로부터의 수정을 암시하는 반면 소문자 `-m`은 그렇지 않습니다. 대신 소문자 `-m`은 편집 프로그램에서 임시 파일을 열고 이 파일에 수행한 모든 변경사항을 저장하고 편집 프로그램를 종료할 때 시스템이 즉시 해당 변경사항을 반영합니다.

그러나 한 번에 많은 오브젝트를 변경하려는 경우나 비 대화식으로 오브젝트 구성을 변경하려는 경우 `qconf -...attr` 명령 세트를 사용하십시오.

하나의 그런 명령 세트는 *파일에 있는* 스펙에 따라서 수정을 수행합니다.

```
qconf -{A,M,R,D}attr queue|exechost|pe|ckpt <파일이름>
```

대응하는 명령 세트는 *명령줄의* 스펙에 따라서 수정을 수행합니다.

```
qconf -{a,m,r,d}attr queue|exechost|pe|ckpt <속성> <값> <대기열목록>|<호스트목록>
```

두 명령 세트 모두에서 옵션은 다음을 표시합니다.

- -A/a - 속성 추가
- -M/m - 속성 수정
- -R/r - 속성 대체
- -D/d - 속성 삭제
- <*속성*> - 변경할 대기열 또는 호스트 속성
- <*값*> - 영향을 받을 속성의 값
- <*파일이름*> - 속성-값 쌍이 들어 있는 파일

a, m 및 d 옵션을 사용하면 값 목록에 있는 개별 값에 대해 동작할 수 있는 반면 r은 값의 전체 목록을 명령줄이나 파일에 지정되는 새 값으로 대체합니다.

*보기*

● **tcf27-e019.q의 대기열 유형을 일괄처리 전용으로 변경하십시오.**

```
% qconf -rattr queue qtype batch tcf27-e019.q
```

● **파일 new.cfg의 내용을 바탕으로 tcf27-e019.q의 대기열 유형과 쉘 시작 작동을 수정하십시오.**

```
% cat new.cfg
qtype batch interactive checkpointing
shell_start_mode unix_behavior
% qconf -Rattr queue new.cfg tcf27-e019.q
```

● **storage 및 license라는 복합을 호스트 tcf27-e019에 추가하십시오.**

```
% qconf -rattr exechost complex_list storage,license tcf27-e019
```

● **값 1000M을 갖는 scratch1 및 값 2를 갖는 long이라는 자원을 추가하십시오.**

```
% qconf -rattr exechost complex_values scratch1=1000M,long=2 tcf27-e019
```

● **short이라는 자원을 값 4를 갖는 호스트에 접속하십시오.**

```
% qconf -aattr exechost complex_values short=4 tcf27-e019
```

● **다른 값은 그대로 두면서 scratch1의 값을 500M으로 변경하십시오.**

```
% qconf -mattr exechost complex_values scratch1=500M tcf27-e019
```

● **자원 long을 삭제하십시오.**

```
% qconf -dattr exechost complex_values long tcf27-e019
```

● **tcf27-b011.q를 체크포인팅 환경 sph에 대한 대기열 목록에 추가하십시오.**

```
% qconf -aattr ckpt queue_list tcf27-b011.q sph
```

● **병렬 환경 make에 있는 슬롯 수를 50으로 변경하십시오.**

```
% qconf -mattr pe slots 50 make
```

## qselect 명령을 사용하여 대기열 대상 지정

qselect 명령은 대기열 목록을 출력합니다. 옵션과 함께 호출되는 경우 주어진 스펙과 일치하는 대기열만 나열합니다. 수정하려는 특정 대기열을 대상으로 하는 qconf -...attr 명령 세트와 조합하여 이 명령을 크게 유리하게 사용할 수 있습니다.

### *보기*

● **Linux 시스템의 모든 대기열을 나열하려면,**

```
% qselect -l arch=glinux
```

● **두 CPU를 갖는 시스템의 모든 대기열을 나열하려면,**

```
% qselect -l num_proc=2
```

● **모든 4 CPU 64 비트 Solaris 시스템의 모든 대기열을 나열하려면,**

```
% qselect -l arch=solaris64,num_proc=4
```

● **응용 프로그램 사용권(이전에 구성된)을 제공하는 대기열을 나열하려면,**

```
% qselect -l app_lic=TRUE
```

qselect를 qconf와 결합하여 단일 명령줄로 광범위한 변경사항을 수행할 수 있습니다. 이렇게 하려면 전체 qselect 명령을 역슬래시(/) 안에 두고 명령줄의 <*대기열목록*> 변수 자리에서 사용하십시오.

*보기*

● **Solaris 시스템의 모든 대기열에서 prolog 스크립트를 sol_prolog.sh로 설정하십시오.**

```
% qconf -mattr queue prolog /usr/local/scripts/sol_prolog.sh ‘qselect -l
arch=solaris‘
```

● **2 프로세스 시스템의 모든 대기열에서 속성 fluent_license를 2로 설정하십시오.**

```
% qconf -mattr queue complex_values fluent_license=2 ‘qselect -l num_proc=2‘
```

qconf 명령과 결합하여 qselect 명령을 사용하면 Sun Grid Engine, Enterprise 버전 5.3 대기열의 구성을 자동화하는 가장 유연한 방법을 제공하여, 사용자 정의 관리 스크립트를 구축할 수 있습니다.

## ▼ 파일을 사용하여 글로벌 구성 또는 스케줄러를 수정하는 방법

- **글로벌 구성을 변경하려면 다음 절의 지침에 따라서 `qconf -mconf` 명령을 사용하십시오.**
- **스케줄러를 변경하려면 다음 절의 지침에 따라서 `qconf -msconf` 명령을 사용하십시오.**

### 논의

이들 명령은 둘 다 편집 프로그램에 임시 파일을 엽니다. 편집 프로그램를 종료할 때 사용자가 이 임시 파일에 저장한 모든 변경사항이 시스템에 의해 처리되고 즉시 적용됩니다. 임시 파일을 여는 데 사용되는 편집 프로그램은 `EDITOR` 환경 변수에 의해 지정되는 프로그램입니다. 이 변수가 정의되지 않은 경우 `vi`가 사용됩니다.

`EDITOR` 환경 변수를 활용하여 `qconf -m...` 명령의 작동을 자동화할 수 있습니다. 이 변수의 값을 파일의 이름이 첫 번째 인수로 제공되는 파일을 수정하는 프로그램을 가리키도록 변경하십시오. 이 프로그램이 임시 파일을 수정하고 종료할 때 시스템이 수정사항을 읽고 즉시 갱신합니다.

---

**참고 –** 파일의 수정 시간이 편집 조작 후에 변경되지 않는 경우 시스템은 가끔 파일이 수정되지 않은 것으로 잘못 가정합니다. 그러므로 다른 수정 시간을 보장하기 위해 파일을 쓰기 전에 `sleep 1` 명령어를 삽입해야 합니다.

---

*예*

다음 예의 스크립트는 스케줄러의 스케줄 간격을 수정합니다.

```
#!/bin/ksh
# sched_int.sh: modify the schedule interval
# usage: sched_int.sh <n>, where <n> is
# the new interval, in seconds. n < 60

TMPFILE=/tmp/sched_int.$$
if [ $MOD_SGE_SCHED_INT ]; then
     grep -v schedule_interval $1 > $TMPFILE
     echo "schedule_interval 0:0:$MOD_SGE_SCHED_INT" >> $TMPFILE
# sleep to ensure modification time changes
     sleep 1
     mv $TMPFILE $1
else
     export EDITOR=$0
     export MOD_SGE_SCHED_INT=$1
     qconf -msconf
fi
```

위의 스크립트는 `EDITOR` 환경이 스스로를 지적하도록 수정한 후 `qconf -msconf` 명령을 호출합니다. 스크립트의 이 두 번째 내포된 호출이 첫 번째 인수에 의해 지정되는 임시 파일을 수정한 후 종료합니다. 그런 다음 Sun Grid Engine, Enterprise 버전 5.3 시스템이 자동으로 변경사항을 읽고 스크립트의 첫 번째 호출이 종료합니다. 위의 기법은 모든 `qconf -m`... 명령과 결합하여 사용할 수 있습니다. 그러나 이것을 자동화하는 다른 방법이 없기 때문에 스케줄러 관리 및 글로벌 구성에 특히 유용합니다.

## ▼ 그리드 환경을 미세 조정하는 방법

Sun Grid Engine, Enterprise 버전 5.3은 전체 기능을 갖는 범용 분산 자원 관리(DRM) 도구입니다. 시스템의 스케줄러 구성요소는 다양한 compute farm 시나리오의 넓은 범위를 지원합니다. 사용자의 연산 환경으로부터 최대 성능을 얻기 위해서는 어떤 기능이 사용 가능하고 부하 관리 문제점을 해결하기 위해 실제로 무엇이 필요한지를 검토하는 것이 바람직할 수 있습니다. 이들 기능의 일부를 사용 불가능하게 하는 것이 클러스터의 처리량에서 성능 이익을 가질 수 있습니다.

## 스케줄러 모니터링

스케줄러 모니터링하면 특정 작업이 작업 지정되지 않은 이유를 찾을 수 있습니다. 그러나 이 정보를 항상 모든 작업에 대해 제공하는 것은 자원 소비적일 수 있으며 대개는 필요하지 않습니다.

- **스케줄러 모니터링을 비활성화하려면 스케줄러 구성 `sched_conf(5)`에서 `schedd_job_info`를 false로 설정하십시오.**

## 종료된 작업

어레이 작업의 경우 `qmaster`에 있는 종료된 작업 목록이 아주 커질 수 있습니다. 이것을 끄면 메모리를 절약하고 `qstat` 프로세스를 가속화할 수 있습니다. `qstat` 프로세스도 종료된 작업 목록을 반입하기 때문입니다.

- **종료된 작업 목록 기능을 끄려면 글로벌 구성 `sge_conf(5)`에서 `finished_jobs`를 `0`으로 설정하십시오.**

## 작업 검증

작업 제출시에 유효성 검증을 강제하는 것은 지정할 수 없는 작업이 보류 중 상태에 영구적으로 남지 못하게 하는 가치있는 절차일 수 있습니다. 그러나 다양한 여러 가지 실행 노드와 소비 가능한 자원을 갖고 모든 사용자가 고유한 작업 프로파일을 갖는 이질적 환경에서는 특히 작업을 유효화하는 것도 시간을 소비하는 업무일 수 있습니다. 소수의 서로 다른 작업만을 갖는 동종 환경에서는 일반 작업 유효성 검증이 대개 생략될 수 있습니다.

- **작업 검증을 비활성화하려면 클러스터측 기본 요청에 `qsub(1)` 옵션인 `-wn`을 추가하십시오(`sge_request(5)` 참조).**

## 부하 임계값 및 일시중단 임계값

시스템을 고의로 과다하게 신청하는 경우 부하 임계값이 필요하며 과다한 시스템 부하를 막기 위한 메커니즘이 필요합니다. 일시중단 임계값도 이를 위해 사용됩니다. 부하 임계값이 필요한 다른 경우는 실행 노드가 여전히 Sun Grid Engine, Enterprise 버전 5.3 시스템의 제어하에 있지 않은 대화식 부하에 대해 열려 있고 사용자가 노드가 과부하되지 않도록 하려는 경우입니다.

compute farm이 게다가 단일 목적인 경우(예를 들어 연산 노드의 각 CPU가 단 하나의 대기열 슬롯에 의해 표시되고 이들 노드에서 대화식 부하가 예상되지 않는 경우) `load_thresholds`를 생략할 수 있습니다.

- **두 임계값을 모두 비활성화하려면 `load_thresholds`를 `none`으로 및 `suspend_thresholds`를 `none`으로 설정하십시오(`queue_conf(5)` 참조).**

## 부하 조정

부하 조정은 작업이 작업 지정된 후 측정된 부하를 가상적으로 증가시키는 데 사용됩니다. 이 메커니즘은 부하 임계값과 정렬하기 위해 과다 신청된 시스템의 경우에 도움이 됩니다. 스케줄러에 호스트 정렬 및 부하 임계값 검증과 연결하여 추가 작업을 부과하기 때문에 필요하지 않은 경우에는 부하 조정을 꺼야 합니다.

- **부하 조정을 비활성화하려면 스케줄러 구성 `sched_conf(5)`에서 `job_load_adjustments`를 `none`으로 및 `load_adjustment_decay_time`을 `0`으로 설정하십시오.**

## 요구시 스케줄링

Sun Grid Engine, Enterprise 버전 5.3 시스템에 대한 기본값은 고정된 스케줄 간격으로 스케줄링 실행을 시작하는 것입니다(`schedd_conf(5)`의 `schedule_interval`을 참조). 고정 간격의 좋은 점은 `qmaster`/`scheduler`의 cpu 시간 소비를 제한한다는 것입니다. 나쁜 점은 스케줄러를 억압하여 인위적으로 제한된 처리량을 생성한다는 점입니다. 많은 compute farm은 `qmaster`/`scheduler`에 특별히 전용되는 시스템을 갖고 있으며 그런 설정에서는 스케줄러를 억압할 이유가 없습니다.

- **글로벌 클러스터 구성 `sge_conf(5)`의 `schedd_params` 섹션에서 `FLUSH_SUBMIT_SEC` 및 `FLUSH_FINISH_SEC` 설정을 사용하여 요구시 스케줄링을 구성하십시오.**

  요구시 스케줄링이 활성화되는 경우 compute farm의 처리량은 `qmaster`/`scheduler`를 호스트하고 있는 시스템의 능력에 의해서만 제한됩니다.

# 정책

Sun Grid Engine, Enterprise 버전 5.3 소프트웨어의 가장 중요한 기능 중 하나는 시스템 관리자가 설정한 정책을 위한 분산 컴퓨터 전원의 개념입니다. 배경 정보는 해당 절, 4 페이지의 “정책, 호스트 및 데몬”을 참조하십시오.

다음 절은 사이트에 대한 기본적인 "기능적" 및 "공유 기반" 정책 구성을 작성하는 방법에 대한 지침을 포함합니다. 추가 기능적 및 공유 기반 설치 뿐 아니라 "기한" 및 "대체" 정책 구성에 대한 지침들은 *Sun Grid Engine, Enterprise 버전 5.3 관리 및 사용 설명서*에 포함되어 있습니다.

## 기능적 정책 구성

이 절의 모든 구성은 기능적 정책을 주 정책으로 사용합니다. 이 구성 유형은 언제라도 각 사용자, 프로젝트 또는 부서에 정의된 공유가 보증된다는 것을 확실하게 합니다. 제안한 것보다 더 적은 자원을 사용해 온 사용자, 프로젝트 또는 부서들의 작업은 시스템이 작업을 신속히 처리하여 자원을 공전시킬 때 선호됩니다.

동시에, 사용되지 않은 공유 재산이 자원을 필요로 하는 해당 사용자, 프로젝트 및 부서 사이에 분배되기 때문에 전체 자원 활용이 보증됩니다. 과거 자원 소비는 계산에 포함되지 않습니다.

### ▼ 사용자 기반 기능적 예약을 작성하는 방법

이 구성의 목적은 다른 사용자에 대해 Sun Grid Engine, Enterprise 버전 5.3 클러스터에 통합된 모든 자원의 특정 공유 할당을 작성하는 것입니다. FCFS 예약은 동일 사용자의 작업 간에 사용됩니다.

1. **전체 구성(`sge_conf(5)`)의 `schedd_params` 절에서 `SHARE_FUNCTIONAL_SHARES=1`을 사용하십시오.**
2. **기능적 티켓의 수를 지정합니다. (예를 들어, 스케줄러 구성(`sched_conf(5)`)의 `1000000`)**
3. **각 예약 관련 사용자에 대해 한 사용자(`user(5)`)를 추가합니다.**
4. **각 사용자에 기능적 공유를 할당합니다.**

   공유를 전체의 백분율로 할당합니다. 예를 들어,

   - 사용자A (10)
   - 사용자B (20)
   - 사용자C (20)
   - 사용자D (50)

### ▼ 프로젝트 기반 기능적 예약을 작성하는 방법

이 구성의 목적은 다른 프로젝트에 대해 Sun Grid Engine, Enterprise 버전 5.3 클러스터에 통합된 모든 자원의 특정 공유 할당을 작성하는 것입니다. FCFS 예약은 동일 프로젝트의 작업 간에 사용됩니다.

1. **전체 구성(`sge_conf(5)`)의 `schedd_params` 절에서 `SHARE_FUNCTIONAL_SHARES=1`을 사용하십시오.**
2. **기능적 티켓의 수를 지정합니다. (예를 들어, 스케줄러 구성(`sched_conf(5)`)의 `1000000`)**

3. **각 예약 관련 프로젝트에 대해 한 프로젝트(`project(5)`)를 추가합니다.**

   acl 및 xacl 기능을 위해 더 높은 특권 프로젝트에 대한 액세스를 제어할 수 있습니다. 이 기능에 대한 정보는 *Sun Grid Engine, Enterprise 버전 5.3 관리 및 사용 설명서* 또는 *Sun Grid Engine 5.3 및 Sun Grid Engine, Enterprise 버전 5.3 참조 설명서*를 참조하십시오.

4. **각 프로젝트에 기능적 공유를 할당합니다.**

   공유를 전체의 백분율로 할당합니다. 예를 들어,

   - 프로젝트A (55)
   - 프로젝트B (45)

## ▼ 부서 기반 기능적 예약을 작성하는 방법

이 구성의 목적은 다른 부서에 대해 Sun Grid Engine, Enterprise 버전 5.3 클러스터에 통합된 모든 자원의 특정 공유 할당을 작성하는 것입니다. FCFS 예약은 동일 부서의 작업 간에 사용됩니다.

1. **전체 구성(`sge_conf(5)`)의 `schedd_params` 절에서 `SHARE_FUNCTIONAL_SHARES=1`을 사용하십시오.**

2. **기능적 티켓의 수를 지정합니다. (예를 들어, 스케줄러 구성(`sched_conf(5)`)의 `1000000`)스케줄러**

3. **각 예약 관련 부서를 추가합니다.**

4. **각 부서에 기능적 공유를 할당합니다.**

   공유를 전체의 백분율로 할당합니다. 예를 들어,

   - 부서A (90)
   - 부서B (5)
   - 부서C (5)

   이 절의 모든 구성은 공유 트리 정책을 주 정책으로 사용합니다. 이 구성 유형은 정의된 공유가 공유트리 초과 시간에 구성된 인스턴스에 보증된다는 것을 확실하게 합니다. 제안한 것(공유트리 공유) 보다 과거에 더 적은 자원을 소비한 공유트리 "브랜치"와 관련된 작업은 시스템이 작업을 신속히 처리하여 자원을 공전시킬 때 선호됩니다. 동시에, 사용하지 않은 공유 재산은 다른 공유트리 브랜치와 관련된 지연 작업에 여전히 사용 가능하기 때문에 전체 자원 활용을 보증합니다.

## ▼ 각 프로젝트 내의 FCFS로 프로젝트 기반 공유트리 예약을 작성하는 방법

이 구성의 목적은 다른 프로젝트에 대해 Sun Grid Engine, Enterprise 버전 5.3 클러스터에 통합된 모든 자원의 특정 공유 할당 시간 초과를 보증하는 것입니다. FCFS 예약은 동일 프로젝트의 작업 간에 사용됩니다.

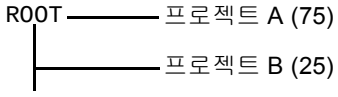
---

**참고 –** 이 문서의 다른 절에 잇는 지침과 달리, 이 절의 일부 지침들은 `qmon` 그래픽 사용자 인터페이스를 사용해야 합니다. `qmon` 사용에 대한 배경 정보 및 지침은 *Sun Grid Engine, Enterprise 버전 5.3 관리 및 사용 설명서*를 참조하십시오.

---

1. **공유트리 티켓의 수를 지정합니다. (예를 들어, 스케줄러 구성(`sched_conf(5)`)의 `1000000`)**
2. **각 예약 관련 프로젝트에 대해 한 프로젝트(`project(5)`)를 추가합니다.**
3. **`qmon` 그래픽 사용자 인터페이스를 사용하여, 노드로서 모든 예약 관련 프로젝트의 구조를 반영하는 공유 트리를 구성하십시오.**
4. **프로젝트에 공유 트리 공유를 할당합니다.**

   단순 구조는 다음 예와 유사합니다.



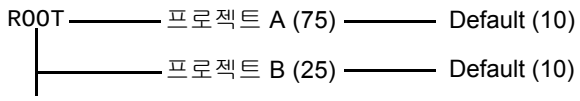

## ▼ 각 사용자에 대해 등가 공유로 프로젝트 기반 공유트리 예약을 작성하는 방법

이 구성의 목적은 다른 프로젝트에 대해 Sun Grid Engine, Enterprise 버전 5.3 클러스터에 통합된 모든 자원의 특정 공유 할당 시간 초과를 보증하는 것입니다. 등가 공유는 동일 프로젝트 내의 자원과 겨루는 작업 사이에서 목표로 정해집니다.

1. **공유트리 티켓의 수를 지정합니다. (예를 들어, 스케줄러 구성(`sched_conf(5)`)의 `1000000`)**
2. **각 예약 관련 사용자에 대해 한 사용자(`user(5)`)를 추가합니다.**
3. **각 예약 관련 프로젝트에 대해 한 프로젝트(`project(5)`)를 추가합니다.**
4. **`qmon` 그래픽 사용자 인터페이스를 사용하여, 노드로서 모든 예약 관련 프로젝트의 구조를 반영하는 공유 트리를 구성하십시오.**

5. **프로젝트에 공유 트리 공유를 할당합니다.**

   단순 구조는 다음 예와 유사합니다.




6. **이러한 각 프로젝트 아래에 해당 트리(나무)의 잎으로서 사용자 `default`를 추가합니다.**

## ▼ 각 프로젝트 내의 사용자 개인당 공유로 프로젝트 기반 공유트리 예약을 작성하는 방법

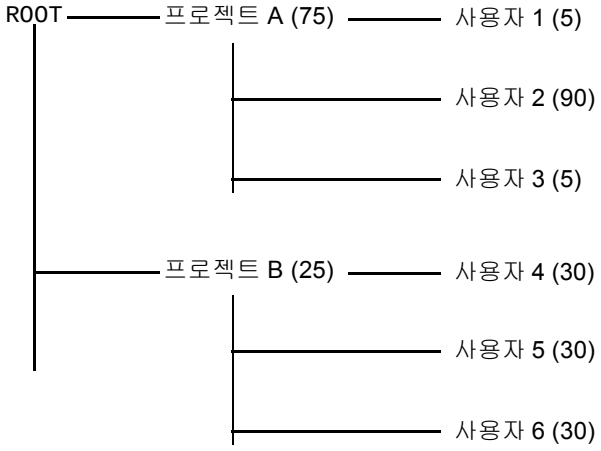
이 구성의 목적은 다른 프로젝트에 대해 Sun Grid Engine, Enterprise 버전 5.3 클러스터에 통합된 모든 자원의 특정 공유 할당 시간 초과를 보증하는 것입니다. 사용자 당 개인 공유가 필요합니다.

1. **공유트리 티켓의 수를 지정합니다. (예를 들어, 스케줄러 구성(`sched_conf(5)`)의 `1000000`)**
2. **각 예약 관련 사용자에 대해 한 사용자(`user(5)`)를 추가합니다.**
3. **각 예약 관련 프로젝트에 대해 한 프로젝트(`project(5)`)를 추가합니다.**
4. **`qmon` 그래픽 사용자 인터페이스를 사용하여, 노드로서 모든 예약 관련 프로젝트의 구조를 반영하는 공유 트리를 구성하십시오.**

5. **프로젝트에 공유 트리 공유를 할당합니다.**

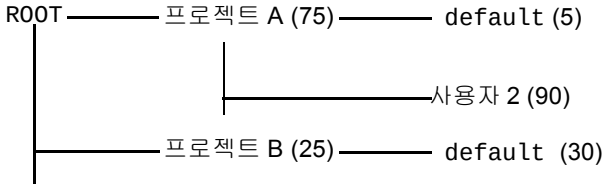
   **액세스 권한이 있고 개인 공유를 할당하는 프로젝트에 있으로서 각 사용자를 추가합니다.**

   단순 구조는 다음 예와 유사합니다.




## 논의

개인 공유 할당이 약간의 사용자들에게만 필요한 경우 프로젝트 노드 아래에 개인 사용자와 결합하여 사용자 `default`를 지정할 수 있습니다. 예를 들어, 위에 묘사된 트리(나무)는 다음으로 요약될 수 있습니다.



# 일반 문제점 해결

다음 절을 사용하여 일반적인 문제점의 원인을 진단하고 응답하는 데 도움을 받으십시오.

- **문제점** - 작업에 대한 출력 파일이 `Warning: no access to tty; thus no job control in this shell...`을 표시합니다.
  - **가능한 원인** - 로그인 파일 중 하나 이상이 `stty` 명령을 포함합니다. 이들 명령은 터미널이 존재하는 경우에만 유용합니다.
  - **가능한 해결책** - Sun Grid Engine, Enterprise 버전 5.3 일괄처리 작업에서 이들 작업과 연관된 터미널이 없습니다. 로그인 파일의 모든 `stty` 명령을 제거하거나 처리하기 전에 터미널을 점검하는 `if` 명령문으로 묶어야 합니다. 다음은 이것의 한 예입니다.

```
/bin/csh:
stty -g            # checks terminal status
if ($status == 0)  # succeeds if a terminal is present
<모든 stty 명령을 여기에 입력합니다>
endif
```

- **문제점** - 작업 표준 오류 로그 파일에 `'tty': Ambiguous`가 있습니다. 그러나 작업 스크립트에서 호출되는 사용자의 쉘에 `tty`에 대한 참조가 없습니다.
  - **가능한 원인** - `shell_start_mode`는 기본적으로 `posix_compliant`입니다. 그러므로 모든 작업 스크립트는 작업 스크립트의 첫 행에서 지정되는 것이 아니라 대기열 정의에 지정되는 쉘과 함께 실행합니다.
  - **가능한 해결책** - `qsub` 명령에 `-S` 플래그를 사용하거나 `shell_start_mode`를 `unix_behavior`로 변경하십시오.
- **문제점** - 명령줄에서 작업 스크립트를 실행할 수 있지만 `qsub` 명령을 통해 실행할 때는 실패합니다.
  - **가능한 원인** - 사용자 작업에 대해 프로세스 한계가 설정되어 있을 수 있습니다. 이것을 테스트하려면 `limit` 및 `limit -h` 기능을 수행하는 테스트 스크립트를 작성하십시오. 쉘 프롬프트에서 대화식으로 및 `qsub` 명령을 통해 둘 다 실행하여 그 결과를 비교하십시오.
  - **가능한 해결책** - 구성 파일에서 사용자 쉘에 한계를 설정하는 모든 명령을 제거하십시오.

■ **문제점** - 실행 호스트가 `99.99`의 로드를 보고합니다.

■ **가능한 원인** - 세 가지 가능성이 존재합니다.

1. `execd` 데몬이 호스트에서 실행되고 있지 않습니다.
2. 기본 도메인이 잘못 지정되어 있습니다.
3. `qmaster` 호스트는 실행 호스트가 그 자체를 참조하는 이름과 다른 것으로 실행 호스트의 이름을 참조합니다.

■ **가능한 해결책** - 원인에 따라 다음 해결책 중 하나를 사용할 수도 있습니다. ("가능한 원인"의 번호를 다음 해결책의 번호에 일치시킵니다.)

1. `$SGE_ROOT/default/common/'rcsge'` 스크립트를 실행하여 실행 호스트에서 `root`로서 `execd` 데몬을 시작하십시오.
2. Sun Grid Engine, Enterprise 버전 관리자로서 `qconf -mconf` 명령을 실행하고 `default_domain` 변수를 `none`으로 변경하십시오.
3. 컴퓨터 클러스터의 호스트 이름을 해결하는 데 DNS를 사용하고 있는 경우, 주 호스트 이름으로 시스템을 나타내는 완전한 이름(FQDN)을 보고하려면 `/etc/hosts` 및 NIS를 구성하십시오. 물론, 여전히 짧은 별명을 정의하여 사용할 수도 있습니다(예를 들어, `168.0.0.1 myhost.dom.com myhost`). DNS를 사용하지 않는 경우에는 모든 `/etc/hosts` 파일 및 NIS 표가 일치하는지 확인합니다(예를 들어, `168.0.0.1 myhost.corp myhost` 또는 `168.0.0.1 myhost`).

■ **문제점** - 30초마다 다음과 비슷한 경고가 <*셀*>`/spool/<host>/messages`에 인쇄됩니다.

```
Tue Jan 23 21:20:46 2001|execd|meta|W|local
configuration meta not defined - using global configuration
```

그러나 각각 `FDQN`을 갖는 파일이 <*셀*>`/common/local_conf/`에 있는 각 호스트에 대해 *존재합니다*.

■ **가능한 원인** - 사용자 시스템에서 해석하는 호스트 이름 `meta`가 단축명을 반환하지만 마스터 시스템에서 `FQDN`을 갖는 `meta`가 반환됩니다.

■ **가능한 해결책** - 모든 `/etc/hosts` 파일과 NIS 표가 이 관계에서 일관성이 있는지 확인하십시오. 이 예에서 `meta` 호스트의 `/etc/hosts` 파일에 다음과 같은 행이 잘못 있을 수 있습니다.

```
168.0.0.1 meta meta.your.domain
```

그러나 행은 대신 다음*이어야* 합니다.

```
168.0.0.1 meta.your.domain meta.
```

■ **문제점** - 가끔 데몬의 `messages` 파일에서 `CHECKSUM ERROR`, `WRITE ERROR` 또는 `READ ERROR` 메시지를 보게 됩니다.

- **가능한 원인** - 이들 메시지가 1초 간격으로 나타나지 않는 한(대개 하루에 1 - 30번 나타날 수 있음) 이 문제에 관하여 어떤 일을 수행할 필요는 없습니다.

■ **문제점** - 작업이 특정 대기열에서 종료하고 qmaster/messages에 다음을 반환합니다.

```
Wed Mar 28 10:57:15 2001|qmaster|masterhost|I|job 490.1
finished on host exechost
```

그러나 그 다음에 실행 호스트의 exechost/messages 파일에서 다음 오류 메시지가 표시됩니다.

```
Wed Mar 28 10:57:15 2001|execd|exechost|E|can't find directory
"active_jobs/490.1" for reaping job 490.1
```

```
Wed Mar 28 10:57:15 2001|execd|exechost|E|can't remove
directory
"active_jobs/490.1": opendir(active_jobs/490.1) failed:
Input/output error
```

- **가능한 원인** - 자동 마운트된 $SGE_ROOT 디렉토리가 마운트 해제되고 있어서 sge_execd 데몬이 그 cwd를 유실하게 합니다.
- **가능한 해결책** - execd 호스트에 대해 로컬 스풀 디렉토리를 사용하십시오. qmon 또는 qconf 명령을 사용하여 매개변수 execd_spool_dir을 설정하십시오.

■ **문제점** - qrsh 유틸리티를 사용하여 대화식 작업을 제출할 때 다음 오류 메시지가 수신됩니다.

```
% qrsh -l mem_free=1G error:error:no suitable queues
```

아직 대기열이 qsub 유틸리티를 사용하는 일괄처리 작업에 사용 가능하며 qhost -l mem_free=1G 및 qstat -f -l mem_free=1G를 사용하여 조회될 수 있습니다.

- **가능한 원인** - 메시지 error:no suitable queues는 -w e submit 옵션의 결과로서, 이 옵션은 qrsh 같은 대화식 작업에 대해 기본적으로 활성화됩니다 (qrsh(1)에서 -w e를 찾으십시오). 이 옵션은 qmaster가 작업이 현재 클러스터 구성에 따라서 작업 지정될 수 있음을 확실히 모르는 경우 submit 명령이 실패하게 합니다. 이 메커니즘의 의도는 작업이 부여될 수 없는 경우에 작업 요청을 미리 거절하는 것입니다.

- **가능한 해결책** - 이 경우, mem_free는 소비 가능한 자원이도록 구성되지만 사용자가 각 호스트에서 사용 가능한 메모리의 양을 지정하지 않았습니다. 메모리 부하 값은 변하기 때문에 클러스터 구성의 일부로 볼 수 없어서 이 점검에 대해 일부러 고려되지는 않습니다. 이것을 극복하기 위해 다음 중 하나를 수행할 수 있습니다.

  -w n과 명시적으로 함께 제출하여 qrsh 기본 설정인 -w e를 대체하여 일반적으로 *이 점검을 생략하십시오*. 또한 이것을 $SGE_ROOT/<*셀*>/common/cod_request에 놓을 수도 있습니다.

  mem_free*를 소비 가능한 자원으로 관리하려는 경우* qconf -me <*호스트이름*>을 사용하여 host_conf(5)의 complex_values에 있는 호스트에 대해 mem_free 용량을 지정하십시오.

  mem_free를 소비 가능한 자원으로 *관리하지 않으려는 경우* qconf -mc *호스트*를 사용하여 complex(5)의 consumable 열에서 다시 소비 불가능 자원으로 만드십시오.

- **문제점** - qrsh가 그것이 존재하는 동일한 노드에 작업 지정하지 않습니다. qsh 쉘에서,

```
host2 [49]% qrsh -inherit host2 hostname
error:executing task of job 1 failed:

host2 [50]% qrsh -inherit host4 hostname
host4
```

  - **가능한 원인** - gid_range가 충분하지 않습니다. 단일 숫자가 아니라 범위로서 정의되어야 합니다. Sun Grid Engine, Enterprise 버전 5.3 시스템은 호스트의 각 작업에 개별 gid를 지정합니다.
  - **가능한 해결책** - qconf -mconf 또는 qmon 그래픽 사용자 인터페이스를 사용하여 gid_range를 조정하십시오. 제안되는 범위는 다음과 같습니다.

```
gid_range                    20000-20100
```

- **문제점** - qrsh -inherit -V가 병렬 작업 안에서 사용될 때는 작동하지 않습니다. 다음 메시지가 수신됩니다.

```
cannot get connection to "qlogin_starter"
```

  - **가능한 원인** - 이 문제점은 내포된 qrsh 호출과 함께 발생하며 -V 스위치 때문입니다. 첫 번째 qrsh -inherit 호출이 환경 변수 TASK_ID(병렬 작업 안에서 밀접하게 통합된 업무의 ID)를 설정합니다. 두 번째 qrsh -inherit 호출은 업무 등록을 위해 이 환경 변수를 사용하며, 이것은 이미 실행 중인 첫 번째 업무와 동일한 ID를 갖고 업무를 시작하려 시도하므로 실패합니다.

- **가능한 해결책** - qrsh -inherit를 호출하기 전에 TASK_ID를 설정 해제하거나 -V 스위치를 사용하지 않고 대신 -v를 사용하고 실제로 필요한 환경 변수만 내보낼 것을 선택할 수 있습니다.

- **문제점** - qrsh가 전혀 작동하지 않는 것으로 보입니다. 다음과 비슷한 메시지가 수신됩니다.

```
host2$ qrsh -verbose hostname
local configuration host2 not defined - using global
configuration
waiting for interactive job to be scheduled ...
Your interactive job 88 has been successfully scheduled.
Establishing /share/gridware/utilbin/solaris64/rsh session to
host exehost ...
rcmd: socket: Permission denied
/share/gridware/utilbin/solaris64/rsh exited with exit code 1
reading exit code from shepherd ...
error: error waiting on socket for client to connect:
Interrupted system call
error: error reading return code of remote command
cleaning up after abnormal exit of
/share/gridware/utilbin/solaris64/rsh
host2$
```

- **가능한 원인** - qrsh에 대한 사용 권한이 적절히 설정되지 않았습니다.
- **가능한 해결책** - $SGE_ROOT/utilbin/에 있는 다음 파일의 사용 권한을 점검하십시오.(rlogin 및 rsh는 root에 의해 setuid되고 소유되어야 함을 유의하십시오.)

```
-r-s--x--x 1 root root 28856 Sep 18 06:00 rlogin*
-r-s--x--x 1 root root 19808 Sep 18 06:00 rsh*
-rwxr-xr-x 1 sgeadmin adm 128160 Sep 18 06:00 rshd*
```

---

**참고 –** $SGE_ROOT 디렉토리도 setuid 옵션을 갖고 NFS 마운트되어야 합니다. 제출 클라이언트로부터 nosuid를 갖고 마운트되는 경우 qrsh(및 연관된 명령)이 작동하지 않습니다.

---

■ **문제점** - 분산 구조를 시작하려고 할 때 qmake가 다음 오류 메시지를 갖고 종료합니다.

```
qrsh_starter:executing child process qmake failed:No such file
or directory
```

- **가능한 원인** - Sun Grid Engine, Enterprise 버전 5.3 시스템은 실행 호스트에서 qmake의 인스턴스를 시작합니다. Sun Grid Engine, Enterprise 버전 5.3 환경(특히 PATH 변수)가 사용자의 쉘 자원 파일(.profile/.cshrc)에서 설정되지 않는 경우 이 qmake 호출이 실패합니다.
- **가능한 해결책** - -v 옵션을 사용하여 PATH 환경 변수를 qmake 작업에 내보내십시오. 전형적인 qmake 호출은 다음과 같습니다.

```
qmake -v PATH -cwd -pe make 2-10 --
```

■ **문제점** - qmake 유틸리티를 사용할 때 다음 오류 메시지가 수신됩니다.

```
waiting for interactive job to be scheduled ...timeout (4 s)
expired while waiting on socket fd 5

Your "qrsh" request could not be scheduled, try again later.
```

- **가능한 원인** - ARCH 환경 변수가 qmake가 호출된 쉘에서 잘못 설정되었을 수 있습니다.
- **가능한 해결책** - ARCH 변수를 사용자 클러스터의 호스트와 일치하는 지원되는 값으로 올바르게 설정하거나 제출시에 올바른 값을 지정하십시오(예: qmake -v ARCH=solaris64 ...).

# 문제점 진단

Sun Grid Engine, Enterprise 버전 5.3 시스템은 문제점 진단을 돕기 위해 여러 가지 보고 방법을 제공합니다. 다음 절은 이들 방법의 사용을 설명합니다.

## 지정되지 않고 있는 보류 작업

때로는 보류 작업이 명백하게 실행될 수 있지만 지정되지 않습니다. 이유를 진단하기 위해 Sun Grid Engine, Enterprise 버전 5.3은 한 쌍의 유틸리티 및 옵션인 `qstat -j` <*작업ID*> 및 `qalter -w v` <*작업ID*>를 제공합니다.

- `qstat -j` <*작업ID*>

  활성화될 때 `qstat -j` <*작업ID*>는 사용자에게 특정 작업이 최종 스케줄링 실행에서 지정되지 않은 이유 목록을 제공합니다. 이 모니터링은 `schedd` 데몬과 `qmaster` 사이의 비적절한 통신 오버헤드를 유발할 수 있기 때문에 활성화 또는 비활성화될 수 있습니다(`sched_conf(5)`의 `schedd_job_info`를 참조하십시오). 다음은 ID `242059`를 갖는 작업에 대한 샘플 출력입니다.

```
% qstat -j 242059
scheduling info: queue "fangorn.q" dropped because it is temporarily not available
                 queue "lolek.q" dropped because it is temporarily not available
                 queue "balrog.q" dropped because it is temporarily not available
                 queue "saruman.q" dropped because it is full
                 cannot run in queue "bilbur.q" because it is not contained in its hard
                 queue list (-q)
                 cannot run in queue "dwain.q" because it is not contained in its hard
                 queue list (-q)
                 has no permission for host "ori"
```

  이 정보는 schedd 데몬이 직접 생성하며 클러스터의 현재 이용율을 고려합니다. 때때로 이것은 정확히 사용자가 원하는 것이 아닙니다. 예를 들어 모든 대기열 슬롯이 이미 다른 사용자의 작업에 의해 점유되는 경우 사용자가 관심을 갖는 작업에 대해 상세한 메시지가 생성되지 않습니다.

- `qalter -w v` <*작업ID*>

이 명령은 작업이 원칙적으로 작업 지정될 수 없는 이유를 나열합니다. 이 목적을 위해 있는 그대로의 스케줄링 실행이 수행됩니다. 이 있는 그대로의 스케줄링 실행에 관해서 특별한 것은 모든 소비 가능한 자원(및 슬롯)이 이 작업에 완전히 사용 가능한 것으로 간주된다는 점입니다. 비슷하게 모든 부하 값은 변하기 때문에 무시됩니다.

## 오류 상태 E로 보고되는 작업 또는 대기열

작업 또는 대기열 오류는 `qstat` 출력에서 대문자 `E`로 표시됩니다. Sun Grid Engine, Enterprise 버전 5.3 시스템이 대기열에 있는 작업을 실행하려 시도하지만 작업에 특정한 이유 때문에 실패할 때 작업은 오류 상태에 들어갑니다. Sun Grid Engine, Enterprise 버전 5.3 시스템이 대기열에 있는 작업을 실행하려 시도하지만 대기열에 특정한 이유 때문에 실패할 때 대기열이 오류 상태에 들어갑니다.

Sun Grid Engine, Enterprise 버전 5.3 시스템은 사용자 및 관리자에게 작업 실행 오류의 경우에 진단 정보를 수집할 기능 세트를 제공합니다. 대기열 및 작업 오류 상태 모두가 실패한 작업 실행에서 나타나므로 진단 가능성은 두 유형의 오류 상태 모두에 적용 가능합니다.

- 사용자 중단 메일

  작업이 `submit` 옵션인 `-m a`와 함께 제출되는 경우 중단 메일이 `-M` *사용자*[`@`*호스트*] 옵션으로 지정된 주소로 전송됩니다. 중단 메일에는 작업 오류에 관한 진단 정보가 들어 있으며 사용자를 위한 권장 정보 소스입니다.

- `qacct` 계정

  중단 메일을 사용할 수 없는 경우 사용자는 `qacct -j` 명령을 실행하여 Sun Grid Engine, Enterprise 버전 5.3 시스템의 작업 계정 기능으로부터 작업 오류에 관한 정보를 얻을 수 있습니다.

- 관리자 중단 메일

  관리자는 적당한 이메일 주소를 지정하여 작업 실행 문제점에 관한 관리자 메일을 주문할 수 있습니다(`sge_conf(5)`의 `administrator_mail` 참조).관리자 메일에는 사용자 중단 메일보다 더 자세한 진단 정보가 들어 있으며 빈번한 작업 실행 오류의 경우에 권장되는 방법입니다.

- 메시지 파일

  관리자 메일을 사용할 수 없는 경우 `qmaster messages` 파일을 먼저 조사해야 합니다.적당한 작업 ID를 검색하여 특정 작업과 관련된 기록을 찾을 수 있습니다. 기본 설치에서 `qmaster messages` 파일은 `$SGE_ROOT/default/spool/qmaster/messages`입니다.

  때로는 작업이 시작된 `execd` 데몬의 메시지에서 추가 정보를 찾을 수 있습니다. `qacct -j` <*작업ID*>를 사용하여 작업이 시작된 호스트를 찾고 작업ID에 대해 `$SGE_ROOT/default/spool/`<*호스트*>`/messages`를 검색하십시오.