国家超级计算广州中心

TH-2 用户手册

试用版

V 1.1

2014

WWW.NSCC-GZ.CN

修订说明

版本	修订人	修订内容	修订日期
1.0	钟英	用户手册试用版	2014-06-05

目　　录

1 TH-2 系统基本环境

1.1 TH-2 系统概述

天河二号超级计算机系统是国家 863 计划和核高基重大专项的标志性成果，是新一代银河高性能计算机关键技术突破的成功应用，由国防科学技术大学研制。广东省和广州市提供了研制经费配套支持，采用天河二号作为广州超级计算中心业务主机。天河二号峰值计算速度每秒 5.49 亿亿次、持续计算速度每秒 3.39 亿亿次、能效比每瓦特 19 亿次，双精度浮点运算。天河二号峰值计算速度、持续计算速度以及综合技术水平处于国际领先地位，2013 年 6 月份，第 41 届国际超级计算机 500 强排名世界第一，是我国超级计算技术发展取得的重大进展。

天河二号由 170 个机柜组成，包括 125 个计算机柜、8 个服务机柜、13 个通信机柜和 24 个存储机柜，占地面积 720 平方米。内存总容量 1.4PB，存储总容量 12.4PB，最大运行功耗 17.8 兆瓦。

1.1.1 硬件环境

天河二号硬件系统由计算阵列、服务阵列、存储子系统、互连通信子系统、监控诊断子系统等五大部分组成。

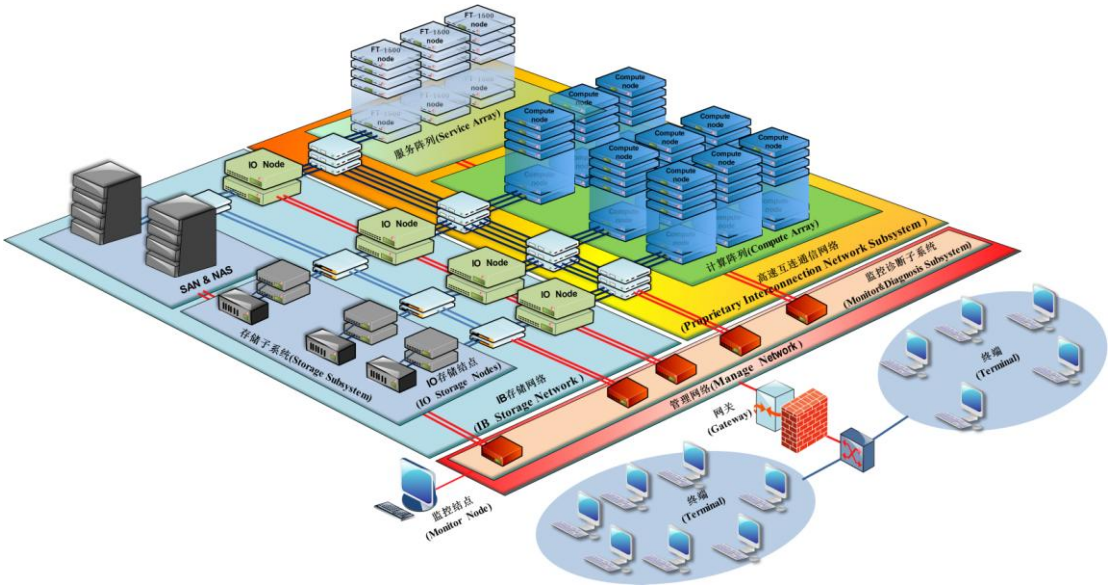

图 1 天河二号硬件系统组成图

1.1.1.1 登录结点 ln[0-5]

LN0-5 登录结点，主要作用是实现用户登录，程序开发，提交任务等工作。
LN0-5 登录结点由 4 路 8 核 Xeon E5-4640 服务器构成，登录结点硬件配置如下：
4 块 Intel Xeon E5-4640 CPU 组成，核心数量 32 核，CPU 主频 2GB；
内存 128GB；

1.1.1.2 计算结点

计算阵列包含 16,000 个计算结点。每个计算结点包含 2 个 Xeon E5 12 核心的多核中央处理器和 3 个 Xeon Phi 57 核心的众核加速器，共 312 万个计算核心。每个结点拥有 64GB 主存，而每个 Xeon Phi 协处理器板载 8GB 内存，故每结点共 88GB 内存，整体总计内存 1.408PB。

1.1.1.3 互连系统

互连通信子系统为自主定制的高速互连系统，采用光电混合技术、胖树拓扑结构、点点带宽 160Gbps，可高效均衡扩展。

1.1.2 软件环境

天河二号软件系统采用高性能计算软件栈架构，由操作系统、文件系统、资源管理系统、编译系统、并行开发工具、应用支撑框架和自治管理系统等构成，形成了系统操作环境、应用开发环境、运行支撑环境和综合管理环境等四大环境。

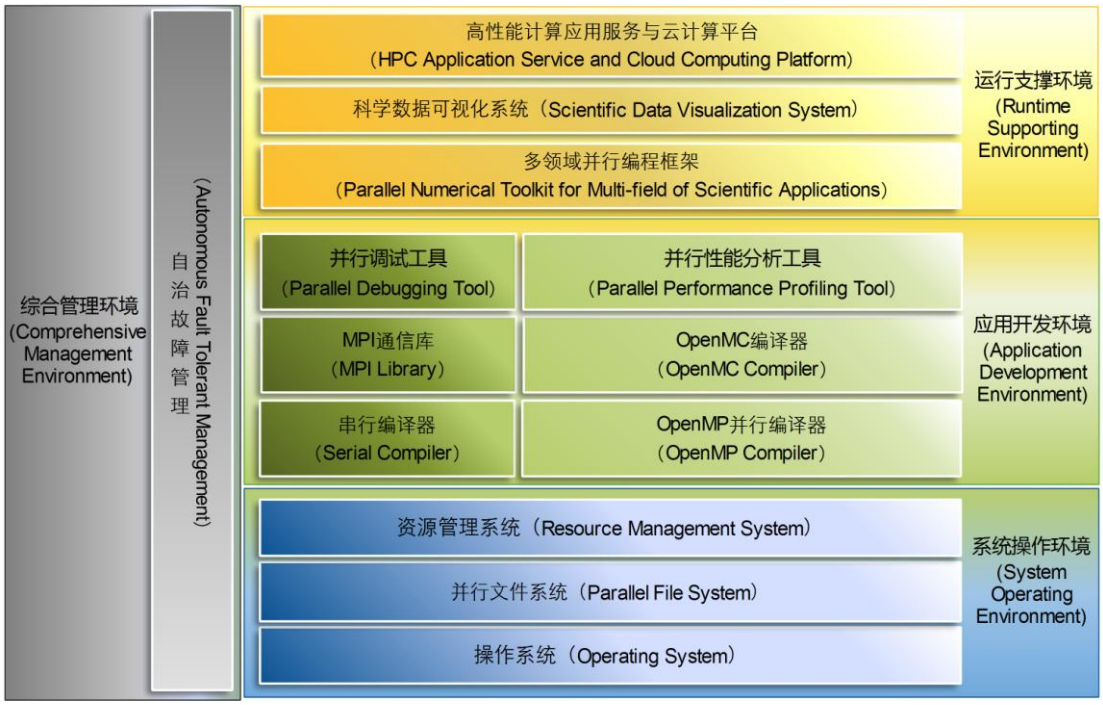

图 2 天河二号高性能计算软件栈架构图

1.2 编译环境

在 TH-2 系统的登录结点 ln0-5 中，已安装好两套完整的编译器系统：一是 Intel 编译器系统，另一是 GCC 编译系统。用户可根据目标程序用途，选择不同的编译系统进行系统和应用程序的开发，由于 TH-2 广泛采用了 Intel 的 CPU，因此在编译中除特定需要，建议首选 Intel 编译器系统。

1.2.1 Intel 编译器

TH-2 系统上安装了 Intel 编译器 11.1、13 和 14 版本，支持 C，C++，Fortran77 和 Fortran90 语言程序的开发。

1.2.1.1 Intel 11.1 编译器

Intel 11.1 编译器的安装路径位于/vol-th/Compiler/11.1/059/目录中，其中：

C 和 C++编译器，以及 Fortran 77/90 的相应命令程序均在：

/vol-th/Compiler/11.1/059/bin/intel64/中，编译命令分别为 icc 和 icpc，ifort 等；

运行所需的 lib 库，在/vol-th/Compiler/11.1/059/lib/intel64/ 下。用户使用时，需导入环境变量 LD_LIBRARY_PATH ：

export LD_LIBRARY_PATH=
/vol-th/Compiler/11.1/059/lib/intel64:$LD_LIBRARY_PATH

Intel 11.1 对应的 mkl 的安装路径为/vol-th/Compiler/11.1/059/mkl，用户可以使用该目录下的 lib/em64t 的 mkl 库。

用户在使用 mkl 库计算任务时，需要设置相应环境变量 LD_LIBRARY_PATH：

export LD_LIBRARY_PATH=/vol-th/Compiler/11.1/059/lib/mkl:$LD_LIBRARY_PATH

1.2.1.2 Intel 13 编译器

Intel 13 编译器的安装路径位于/vol-th/intel/composer_xe_2013.0.079/目录中，其中：

C 和 C++编译器，以及 Fortran 77/90 的相应命令程序均在：

/vol-th/intel/composer_xe_2013.0.079/bin/intel64/中，编译命令分别为 icc 和 icpc，ifort 等；

运行所需的 lib 库，在/vol-th/intel/composer_xe_2013.0.079/lib/intel64/ 下。用户使用时，需导入环境变量 LD_LIBRARY_PATH ：

export LD_LIBRARY_PATH=
/vol-th/intel/composer_xe_2013.0.079/lib/intel64:$LD_LIBRARY_PATH

Intel 13 对应的 mkl 的安装路径为/vol-th/intel/composer_xe_2013.0.079/mkl，用户可以使用该目录下的 lib/em64t 的 mkl 库。

用户在使用 mkl 库计算任务时，需要设置相应环境变量 LD_LIBRARY_PATH：

export LD_LIBRARY_PATH=
/vol-th/intel/composer_xe_2013.0.079/lib/mkl:$LD_LIBRARY_PATH

1.2.1.3 Intel 14 编译器

Intel 14 编译器的安装路径位于/vol-th/intel/composer_xe_2013_sp1.1.106/目录中，其中：

C 和 C++编译器，以及 Fortran 77/90 的相应命令程序均在：

/vol-th/intel/composer_xe_2013_sp1.1.106/bin/intel64/中，编译命令分别为 icc 和 icpc，ifort 等；

运行所需的 lib 库，在/vol-th/intel/composer_xe_2013_sp1.1.106/lib/intel64/下。用户使用时，需导入环境变量 LD_LIBRARY_PATH ：

export LD_LIBRARY_PATH=
/vol-th/intel/composer_xe_2013_sp1.1.106/lib/intel64:$LD_LIBRARY_PATH

Intel 11.1 对应的 mkl 的安装路径为/vol-th/intel/composer_xe_2013_sp1.1.106/mkl，用户可以使用该目录下的 lib/em64t 的 mkl 库。

用户在使用 mkl 库计算任务时，需要设置相应环境变量 LD_LIBRARY_PATH：

export LD_LIBRARY_PATH=
/vol-th/intel/composer_xe_2013_sp1.1.106/lib/mkl:$LD_LIBRARY_PATH

注意：

1. 用户默认环境变量 PATH 已经设置包含了/vol-th/intel/composer_xe_2013_sp1.1.106/bin/intel64/ ，用户可以直接使用 icc，icpc，ifort 等进行编译。

2. LN0-5 为登录和编译结点，为完整的操作系统，而计算结点为了保证计算效率，安装的为精简操作系统，所以用户运行时需要指定动态链接库为共享文件系统下的目录/vol-th/lib64/，在此目录下指定需要加载的动态链接库。

常用编译选项如下：

（1） 优化选项

-O0：禁止优化

-O1：优化代码大小和代码局部性。

-O2（缺省值）：优化代码速度（推荐使用）

-O3：-O2+激进的优化（循环、存储访问转换、预取）。需要注意的是，-O3 并不一定适合所有程序。

-fast：打开-O3、-ipo、-static、-no-prec-div 和 -xP

-ipo：过程间优化

（2） 输出和调试选项

-c：只生成目标文件

-S：只生成汇编文件

-g：调试选项

-o <file>：指定生成的输出文件名

（3） 浮点选项

-mp：维持浮点精度（禁止某些优化）

-mp1：改善浮点精度。和-mp 相比，-mp1 对性能影响较小

（4） 链接选项

-L<dir>：指定链接时搜索的库路径

-l<string>：链接特定库

-static：静态链接

-shared：生成共享库

1.2.2 GCC 编译器

TH-2 上默认安装的 GCC 版本是 4.4.6，相关的编译命令都安装到/usr/bin 目录中。

1.2.3 MPI 编译环境

由于 TH-2 采用了自主互连的高速网络，因此底层 MPI 为自主实现，基于 Intel 编译器进行编译。

基于 Intel 编译器的 mpi 版本安装目录在/usr/loca/mpi3 下，为了追求最高效率，该目录下的 mpi 为自主实现的 mpi 版本，底层用 Intel 编译器编译。 基本使用时（运行程序没特殊要求时）推荐使用/usr/local/mpi3 版本，有较高的效率。

并行 mpi 编译环境使用注意事项：

1. TH-2 安装了自主实现的 mpi，程序如无特殊需要，推荐使用/usr/local/mpi3 目录下的 mpi。该 mpi 调用 Intel 14 编译器，且该 mpi 的库均为静态库，用户不用担心动态链接库问题。

2. TH-2 具备自主高速互联网络，并提供 MPI 编程环境，如用户必须使用其他版本 mpi，

比如 openmpi1.4.8，mpich2-1.3.1 等，也可以自己安装并部署。用自行 mpi 编译的程序，同样可以利用高速互联网络的虚拟以太网运算任务，但性能会较 TH-2 自主 MPI 低很多。

MPI 编译命令内部会自动包含 MPI 标准头文件所在的路径，并自动连接所需的 MPI 通信接口库，所以不需要用户在命令行参数中指定。

如果用户使用 makefile 或 autoconf 编译 MPI 并行程序，还可以将 makefile 中的 CC，CXX，F77，F90 等变量设置成 mpicc，mpicxx，mpif77，mpif90，或这在 autoconf 的 configure 过程前设置 CC，CXX，F77 和 F90 等环境变量为 mpicc，mpicxx，mpif77 和 mpif90 等。

1.2.4 MIC 编译环境

TH-2 每个计算结点配置了 3 个 Xeon Phi 57 核心的众核加速器，因此 LN0-5 中，有相应的 MIC 编译环境。

MIC 编译器包含 Intel 13 和 14 两个版本编译器，目前支持 native 和 offload 两种编程模式。

目前用户环境默认支持 Intel 14 编译器，若需 13 版编译器，需做以下操作：

```
source
/vol-th/home/testuser1/intel/composer_xe_2013.3.163/bin/compilervars.sh intel64
```

注意：目前用户若需登陆 MIC 使用 native 模式计算，需要计算结点的 root 权限，该权限使用需申请。

2 TH-2 使用方式

为了更好的保证用户的数据安全，在用户使用中心资源前，需要具备如下条件：

2.1 基本条件

用户需要具备的基本条件如下：

1. 经过了中心用户基本审查创建流程，并填写了相应的文件和协议。

2. 具备一个 VPN 账号及密码。（仅远程连接用户需要）

3. 具备一个系统用户及密码。

具备上了上述条件，您就可以尝试登陆至 TH-2 使用系统资源，登陆 TH-2 的步骤及所需软件如下节描述。

2.2 登录和数据传输

2.2.1 VPN 远程连接

windows 系统的 VPN 远程连接方式登陆 TH-2 步骤如下：

1）使用 IE 浏览器打开登陆地址 https://61.144.43.67:6443/，点击网页右上角有个“SSLVPN 文件下载“，下载并安装运行；

2）使用 IE 浏览器打开登陆地址 https://61.144.43.67:6443/，输入 VPN 账号和密码，保持网页打开状态；

3）使用 ssh 客户端软件（如 SSH Secure Shell Client，SecureCRT，Putty）连接 172.16.21.103，输入用户账号和密码，即可远程使用。SSH Secure Shell Client，SecureCRT，Putty 等均为免费软件，网络上均有下载。

2.2.2 非远程连接方式

同时客户还可以选择非远程连接方式，即到中心上机。TH-2 提供了 ln0-ln5 六个登录结点，登陆 IP 地址，上机时相关工作人员会告知。使用 ssh 客户端软件，输入登录结点 IP 地址、账户名和密码，即可登录至登录服务结点。之后，用户即可以开始编译、提交任务等操作。

特别注意：

TH-2 的 LN0-5 为登录服务结点，只负责用户的登录、编译、提交任务等操作，不允许直接在 LN0-5 运行可执行程序。同时，ln5 为登陆跳转结点，不可提交作业，用户登陆后请跳转其他登录结点进行操作。跳转命令为：“ssh ”+服务结点名，如“ssh ln0”。

2.2.3 文件传输

从外部机器向 TH-2 中上传文件，可以使用 sftp 客户端，例如 SSH Secure Shell Client 等本身自带的文件传输功能，或者使用 WinScp 的 sftp 数据传输软件（免费软件，网络容易下载，且该软件支持断点续传，推荐使用）。

以上软件即可实现文件传输功能，推荐使用 WinSCP 传输软件。

注意：

TH-2 中的存储空间只作为数据的临时存储，请用户及时把重要数据或敏感数据保存到自己的计算机中，并及时清理自己的存储空间，中心不对系统中的任何数据丢失或传播负责。

2.3 环境变量设置

根据用户帐号使用的 Shell 的不同，设置环境变量的方法也有所不同。假设我们要增加一个用来表示字符串“/usr/local/bin”的环境变量 MYENV，可以采用下面的方法来设置。TH-2 默认用户选择的环境变量为 Bash。

1）Bash 的设置方法

```
export MYENV=/usr/local/bin
```

如果需要环境变量在登录进用户帐号后自动设置，则可以编辑用户帐号起始目录

（$HOME）下的.bashrc 文件，将上述命令行加入文件中。

2）sh 的设置方法

```
MYENV=/usr/local/bin
export MYENV
```

如果需要环境变量在登录进用户帐号后被自动设置，则编辑用户帐号起始目录（$HOME）下的.bashrc_profile 文件，将上述命令行加入文件中。

2.4 退出系统

执行“exit”命令或按“ctrl-d”键，即可退出系统。

2.5 用户账号密码修改

目前系统采用了 LDAP 技术，来管理用户，新创建的用户第一次登陆服务结点时会创建相应的工作目录。用户可以通过 passwd 命令修改用户密码，以 customer 用户为例，举例说明如下：

```
[customer@ln0 ~]$ passwd
Changing password for user customer.
Enter login(LDAP) password:
New password:
Re-enter new password:
LDAP password information changed for customer
passwd: all authentication tokens updated successfully.
```

首先需要输入中心给分配的账户密码，之后再输入新的密码，重复输入一次后，就会显示密码更新成功。

特别提示：

为了保证您用户的数据安全，中心采用了多种方法和技术手段，但您也需要保证您的系统用户密码不外泄，希望您能经常更换系统用户密码（两个月更换一次为宜）。

3 TH-2 作业提交

在 TH-2 中，所有在计算结点中运行的串行或并行应用程序，都必须通过资源管理系统来提交运行。资源管理系统首先将用户提交的应用程序构造成作业进行排队处理，然后根据 TH-2 的实时运行资源状态，决定何时以及在哪些计算结点中加载应用程序的运行，不同的应用程序之间不存在资源的竞争冲突，用户也可以通过作业管理系统来监控应用程序的运行。

3.1 使用限制

但为了保证系统资源的高效使用，用户请求的快速响应，系统的稳定性，在系统中做出了相应的使用限制，相关限制如下：

3.1.1 分区限制

TH-2 可根据用户的使用情况，对所有计算资源进行分区。不同分区针对不同的用户群体开放使用，用户可以使用使用 yhi -l 命令，看到相应的分区限制信息。

其中 PARTITION 表示分区，TIMELIMIT 表示该分区的时间限制，NODES 表示结点数，STATE 表示结点运行状态其中 down 表示未启动，idle 表示启动后处于空闲状态，allocated 表示结点已经分配了一个或多个作业，NODELIST 为结点列表。

所有分区均可以设定相应允许的用户队列，目前分区为所有用户均可以使用；中心根据用户的不同分类，划分不同的资源，用户如果看不到某些分区，是因为该用户不具备相应的资源使用权限。

注意：

1. 由于大型集群系统具备一定故障率，TH-2 十分庞大，为了保证系统稳定性，分区中有限定任务执行时间的限制，因此建议用户为程序设立“断点”从而保证任务由于意外中断后，可以继续运算。

2. 如果用户的程序没有办法“续算”，而且运行时间超过 48 小时，请联系中心技术人员。

3. 目前 TH-2 对所有用户仅设一个分区，所有用户共享该分区资源。若用户需要独占资源，请提出申请。同时分区后续若有变动，该手册会及时更新。

3.1.2 用户限制

除了上述的分区限制，目前还根据用户的申请情况，针对用户做了一定的限制，用户可以使用：

yhacctmgr list user witha

命令来查看自己的限制情况，例如针对用户 fenjh，则可看到相应的限制包括：

maxnodes：最多可以使用的结点数，为 16。

maxjobs：最多可以运行的作业数，为 3。

maxsubmit：包括正在运行和等待的作业总数上限，为 3。

用户同样可以查看到自己的限制，也可以根据后面的需要向中心提出申请，中心会根据用户需要重新修改限制。

为了保证系统和用户数据的安全，目前普通用户不能在没有申请资源时，就 ssh 链接到计算结点，只有分配了相应的计算结点资源后，才能 ssh 到指定计算结点。

3.2 状态查看命令

在用户提交作业前，应查看系统的使用情况，这样利于用户根据系统使用情况，对相应的计算结点进行选择。

3.2.1 结点状态查看 yhinfo 或 yhi

yhi 为 yhinfo 命令的简写，用户可以使用 yhi 或者 yhinfo 命令查看结点的使用情况，从而根据情况做出选择。

其中 PARTITION 表示分区,TIMELIMIT 表示该分区的时间限制,NODES 表示结点数,STATE 表示结点运行状态其中 down 表示未启动，idle 表示启动后出于空闲状态，allocated 表示结点已经分配了一个或多个作业，NODELIST 为结点列表。

3.2.2 作业状态信息查看 yhqueue 或 yhq

yhq 为 yhueue 命令的简写，用户可以使用 yhq 或 yhqueue 命令查看系统中各计算结点的运行情况。

其中 JOBID 表示任务 ID，Name 表示任务名称，USER 为用户，TIME 为已运行时间，NODES 表示占用结点数，NODELIST 为任务运行的结点列表。获取的 jobid，用户在作业取消命令 yhcancel 中会使用到。

用户可以使用 yhq 查看自己提交的作业，为了保证用户的数据安全，普通用户通过 yhq 只能看到自己提交的作业。

3.3 提交作业

目前 TH-2 部署的资源管理系统包括多种作业提交方式，交互作业提交方式 yhrun，批处理作业提交方式 yhbatch 和分配模式 yhalloc。作业终止方式为 yhcancel 命令，需要获取作业的 jobid，如前所述，jobid 可以通过 yhq 命令查看获得。

本手册，为了简化和方便用户，只对相关命令做简单介绍，用户如需更多参数选择，则可以通过相应命令后加入--help 的方式，获取帮助信息，从而满足用户需求。

3.3.1 交互式作业提交 yhrun

系统中作业的运行分成两步：资源分配与任务加载。对于批处理作业，使用 yhbatch 命令提交作业脚本，作业被调度运行后，在所分配的首个结点上执行作业脚本，在作业脚本中使用 yhrun 命令加载作业任务。对于交互式作业，资源分配与任务加载两步均通过 yhrun 命令进行：当在登录 shell 中执行 yhrun 命令时，yhrun 首先向系统提交作业请求并等待资源分配，然后在所分配的结点上加载作业任务。

yhrun 运行的主要格式如下：

yhrun [options] program

yhrun 包括多个选项，用户最常使用的选项如下：

-n, --ntasks=ntasks

指定要运行的进程数。请求 yhrun 分配/加载 ntasks 个进程。省缺的情况是每个 CPU 运行一个进程，但是 -c 参数将改变此省缺值。

-N, --nodes=minnodes[-maxnodes]

请求为此作业至少分配 minnodes 个结点。调度器可能决定在多于 minnodes 个结点上启动作业。可以通过指定 maxnodes 限制最多分配的结点数，如“--nodes=2-4”。最少和最多结点数可以相同以便指定确切的结点数，如“--nodes=2-2”将请求两个并且仅仅两个结点。如果没有指定-N，省缺的行为是分配足够的结点以满足-n 选项的要求。

-p, --partition=partition

从分区 partition 请求资源。如未指定，则省缺为默认分区。

-t, --time=minutes

设置作业的运行时间限制为 minutes 分钟。省缺值为分区的时间限制值。当到达时间限制时，作业的进程将被发送 SIGTERM 以及 SIGKILL 信号终止执行。

-D, --chdir=path

加载的作业进程在执行前将工作目录改变到 path 。省缺情况下作业 yhrun 进程的当前工作目录。

-l, --label

在标准输出/标准错误的每行之前添加任务号。通常，远程任务的标准输出和标准错误通过行缓冲直接传递到 yhrun 的标准输出和标准错误。--label 选项将在每行输出前面添加远程任务的 ID。

-J, --job-name=jobname

指定作业的名字。省缺值是可执行程序的名字 program 。

-W, --wait=seconds

指定在第一个任务退出后，到终止所有剩余任务之前的等待时间。0 表示无限等待（60 秒后将发出一个警告）。省缺值可由系统配置文件中的参数设置。此选项用于确保作业在一个或多个任务提前退出时能够及时终止。

-w, --nodelist=nodelist|filename

请求指定列表中的结点。分配给作业的将至少包含这些结点。nodelist 可以是逗号分割的结点列表或范围表达式（如 cn[1-5,7,12]）。如果包含“/”字符，则 nodelist 将会被当作是一个文件名，其中包含了所请求的结点列表。

-x, --execlude=nodelist|filename

排除指定列表中的结点。分配给作业的将不会包含这些结点。

--checkpoint-path=path

指定任务检查点映像文件的保存目录。省缺为任务的当前工作目录。

--checkpoint-period=number[h|m]

指定对作业进行自动周期性检查点操作。如果 number 后没有跟时间单位，则默认为 h（小时）。

--restart-path=path

指定本次任务加载为从以前的检查点映像恢复执行。path 为检查点映像文件所在的路径。

--exclusive

此作业不能与其它运行的作业共享结点，加入此选项，则表示用户需要针对此作业使用独占的处理器，如果没有足够的处理器，则作业的启动将会被推迟。

以上选项中，由以 -N ，-n，-p，-w，-x 等选项最常用，-N 指定结点数，-n 指定进程数，-p 指定分区名，-w 指定结点列表，-x 指定不参加分配的结点列表（用于排除自己认为有问题的结点）。

TH-2 上的资源使用非抢占式调度方式，即作业如果没有占满结点，则如有别的作业提出需求，若剩余资源合适，也会将资源分配给新的作业。例如一个作业占用了一个结点的 4 核，另外有新的作业也需要 4 核，则该作业也会分配在该结点上。

示例：

1）在分区 corpor，结点 cn[256-268] 上运行 hostname；

```
$ yhrun -w cn[256-268] -p corpor hostname
yhrun: XXXXX: use ‘-t’ option to set time limit of job. defaults to 5 (minutes)
yhrun: job 9637 queued and waiting for resources
yhrun: job 9637 has been allocated resources
cn256
cn259
...
cn267
```

2）运行在 tryout 分区，运行 4 任务的 MPI 程序 cg.C.4，每个结点一个任务，分配的

结点中至少包含结点 cn[4-5]；作业运行时间不超过 20 分钟；

```
$ yhrun -w cn[4-5] -n 4 -N 4 -t 20 ./cg.C.4
NAS Parallel Benchmarks 3.2 --CG Benchmark
Size: 150000
Iterations: 75
Number of active processes: 4
Number of nonzeroes per row: 15
Eigenvalue shift: .110E+03
iteration ||r|| zeta
1 0.15244429457374E-12 109.9994423237398
2 0.45529118072694E-15 27.3920437146522
3 0.45039339889198E-15 28.0339761840269
4 0.44936453849220E-15 28.4191507551292
yhrun: interrupt (one more within 1 sec to abort)
yhrun: task[0-3]: running
5 0.44884028024712E-15 28.6471670038895
6 0.44551302644602E-15 28.7812969418413
```

特别注意：

1. yhrun 基本可以替代 mpirun，特别是使用/usr/local/mpi3 目录下 mpi 编译的程序，完全可以使用 yhrun 提交任务，而不需使用 mpirun。

2. yhrun 为交互式作业提交方式，用户如需要和程序进行交互，则选择直接使用 yhrun 提交任务，如果不需要交互，则需使用批处理作业提交方式。

3. yhrun 提交的任务，如果没有进行输入输出的重定向，在关闭登陆客户端软件时，会导致任务中断，因此如无特殊需要，请直接使用 yhrun 提交任务时，重定向输入输出，并保留相应的 log 文件，方便遇到问题时，技术人员及时解决。

重定向举例如下：

```
yhrun -p test -N 16 -n 128 ./a.out >log 2>&1 &
```

>为重定向符号，2>&1 表示标准错误输出重定向至标准输出，最后的&表示后台提交方式，这样保证了该任务在登陆客户端关闭时依然保持不中断。

4. 再次提示，为了保证任务的稳定性，如无特殊需要请使用批处理作业提交方式。

3.3.2 批处理作业 yhbatch

由于交互需求，才考虑直接使用 yhrun 提交任务。如无交互需求，或不能直接使用 yhrun 提交任务，请使用批处理作业提交任务。

批处理作业是指用户编写作业脚本，指定资源需求约束，然后作为作业提交。提交批处理作业的命令为 yhbatch，用户提交命令后即执行结束，返回命令行窗口，但此时作业在进行排队调度，在资源需求被满足是，分配完计算结点之后，系统将在所分配的第一个计算结点上加载执行用户的作业脚本。

批处理作业使用 yhbatch 命令提交，用户在 yhbatch 的参数中指定资源分配的需求约束，编写的作业脚本中，也可以使用 yhrun 命令加载计算作业，此时 yhrun 通过环境变量感知已经分配了资源，从而直接创建作业而不再次提交作业。

批处理作业的脚本为一个文本文件，脚本第一行以“#!”字符开头，并制定脚本文件的解释程序，如 sh，bash，rsh ，csh 等。

这种作业提交方式，适合那些需要指定资源，且带有自己执行命令的计算作业，或者需要连续执行多个任务的作业，用户可以在脚本中提交多个任务，逐个计算。

如前所述，系统中作业的运行分成两步：资源分配与任务加载。批处理作业使用 yhbatch 提交脚本的方式运行，yhbatch 负责资源分配，yhbatch 获取资源后，会在获取资源的第一个结点运行提交的脚本。

举例一如下：

用户的脚本为 mybash.sh 如下：

```
#! /bin/bash

yhrun  -n 16  -p tryput  -w cn[0-1] hostname
```

根据该脚本用户提交批处理作业，需要明确申请的资源为 tryout 分区的结点 cn[0-1]，因此用户提交如下的批处理命令即可：

```
yhbatch  -w cn[0-1]  -p tryout ./mybash.sh
```

此时注意，给文本文件可执行权限，利用 chmod 命令，使用为：chmod +x filename（其中 filename 替换为你需要修改的文件名。）

计算完成后，工作目录中会生成以 slurm 开头的.out 文件为输出文件。

yhbatch 包含多个选项，基本和 yhrun 类似，用户可以通过 yhbatch --help 命令查看相应所需参数。

举例二 ：

yhbatch 提交的脚本中即可以包含 yhrun，也可以支持 mpirun 等提交作业方式。例如使用了/usr/lib64/openmpi/1.4-gcc 目录下的 openmpi 编译生成可执行程序 a.out，需要运行在结点 cn12-cn27，共计 16 个结点 128 个进程。则安装 mpirun 提交任务的规则，需要撰写 hostlist 文件包含 cn12-cn27，如下所示：

```
cn12:8
cn13:8
cn14:8
cn15:8
cn16:8
cn17:8
cn18:8
cn19:8
cn20:8
cn21:8
cn22:8
cn23:8
cn24:8
cn25:8
cn26:8
cn27:8
```

之后撰写脚本 sub.sh 如下：

```
#!/bin/bash
/usr/lib64/openmpi/1.4-gcc/bin/mpirun  -hostfile hostlist  -np 128 ./a.out
```

用户根据该脚本（chmod 修改该脚本可执行权限 chmod +x sub.sh），提交批处理命令如下：

```
yhbatch  -N 16  -p test  -w cn[12-27] ./sub.sh
```

特别提示：

批处理作业提交模式，试用范围很广，由于手册篇幅限制，不能详述，如果您在提交批处理作业的过程中遇到了任何问题，请联系中心技术人员。

3.3.3 分配模式作业 yhalloc

分配作业模式类似于，交互式作业模式和批处理作业模式的融合。用户需要指定资源分配的需求条件，向资源管理器提出作业的资源分配请求。作业排队，当用户请求资源被满足时，将在用户提交作业的结点上，执行用户所指定的命令，指定的命令执行结束后，也运行结束，用户申请的资源被释放。

yhalloc 后面如果没有跟定相应的脚本或可执行文件，则默认选择了/bin/sh，用户获得了一个合适环境变量的 shell 环境。

yhalloc 和 yhbatch 最主要的区别是，yhalloc 命令资源请求被满足时，直接在提交作业的结点执行相应任务。而 yhbatch 则当资源请求被满足时，在分配的第一个结点上执行相应任务。

yhalloc 在分配资源后，再执行相应的任务，很适合需要指定运行结点，和其它资源限制，并有特定命令的作业。例如 ansys 或其他工程仿真软件的模块，以 ansys 的 lsdyna 模块为例，在并行计算机系统中，lsdyna12.1 版本，需要指定相应的 memory，相应的执行结点列表。由于用户需要在命令中指定相应计算结点，则适合用 yhalloc。

例如：ansys 用户需要 8 个结点，32 个进程，每个结点 4 核的计算资源，利用

yhalloc，有两种提交方式。

第一种首先申请资源，执行如下命令：

```
yhalloc  -N 8  -n 32
```

通过 yhq 查看相应的 jobID 为 163，结点为 cn[50-57],则用户可以选择如下方式：

```
ssh cn50
```

切换到 cn50 结点，之后执行如下命令：

```
lsdyna121  pr=dyna  -dis  memory=250m  i=test.k  o=test.out \
 -machines  cn60:4:cn61:4:62:4:63:4:64:4:65:4:66:4:67:4
```

则可以正常执行 lsdyna 程序。

第二种作业提交方式：

首先通过 yhi 命令，查看哪些结点空闲，确定 8 个空闲的结点，如确定的 8 个空闲结点为 cn[54-61]，则写如下脚本 lsdyna.sh：

```
#!/bin/bash
lsdyna121  pr=dyna  -dis  memory=250m  i=test.k  o=test.out \
 -machines  cn64:4:cn65:4:66:4:67:4:68:4:69:4:70:4:71:4
```

然后执行如下命令：

```
yhalloc  -N 8  -n 32  -w cn[54-61] ./lsdyna.sh
```

使用如上方式，请注意，通过 chmod +x lsdyna.sh 给脚本加可执行权限。

yhalloc 包含多个选项，基本和 yhrun 类似，用户可以通过 yhalloc --help 命令查看相应所需参数。

特别提示：

1. yhalloc 和 yhbatch 的使用方法类似，主要区别为任务加载点不同，yhalloc 命令资源请求被满足时，直接在提交作业的结点执行相应任务。而 yhbatch 则当资源请求被满足时，在分配的第一个结点上执行相应任务。

2. yhalloc 提交的作业，如果需要关闭客户端，请重定向输入输出，并后台提交，可参考 3.3.1 小节的特别提示第三条。

3.4 任务取消 yhcancel

用户可以使用 yhcancel 命令取消自己的作业或作业步。命令格式如下：

```
yhcancel  jobid
```

对于排队作业，取消作业将简单地把作业标记为 CANCELLED 状态而结束作业。对于运行中或挂起的作业，取消作业将终止作业的所有作业步，包括批处理作业脚本，将作业标记为 CANCELLED 状态，并回收分配给作业的结点。一般地，批处理作业将会马上终止；交互作业的 yhrun 进程将会感知到任务的退出而终止；分配模式作业的 yhalloc 进程不会自动退出，除非作业所执行的用户命令因作业或任务的结束而终止。但是在作业被取消时，控制进程都会发送通知消息给分配资源的 yhrun 或 yhalloc 进程。用户可以选择通过 yhalloc 的

--kill-command 选项设置在收到通知时向所执行的命令发送信号将其终止。

3.5 备注

由于手册篇幅限制，只列出了对于绝大多是用户比较重要的相关内容，如您有其他需求也可以联系中心技术人员。

重点提示：不要在登陆结点直接运行可执行程序（极大的影响其他用户的登陆和使用效率）；如无特殊需要，请使用批处理方式提交任务，如果有任何问题请联系中心技术人员；请保存好运行程序的 log 文件，从而方便中心技术人员在作业出问题后的解决问题；若需登陆计算结点运行程序，需要先分配计算结点，方可登陆。