Parallel Processing in the Utah WSC

Ucode Parallel Processing Module

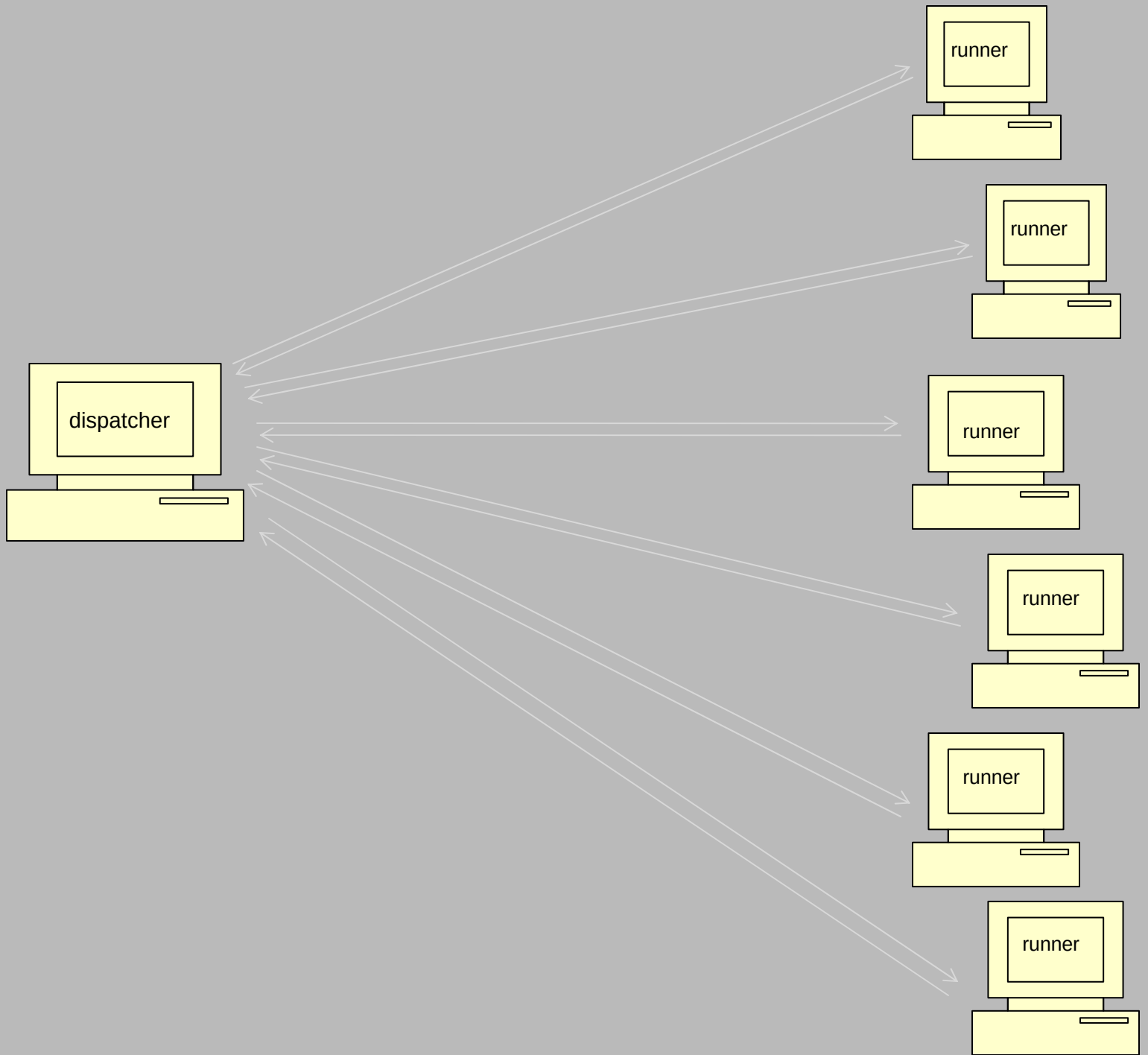
The Utah WSC is using UCODE_2005 in the development of the Great Basin Model to gain insight into parameter sensitivity, correlation, and the effects of potential changes to the conceptual flow model.

Ucode is a universal inverse modeling code created by the U.S. Dept. of Defense, International Ground water Modeling Center, and the U.S. Geological Survery.

Any application model can be used, the only requirement is that ascii input and output files are used and numbers in the files have sufficient significant digits.

Ucode is distributed with a module that offers parallel processing capabilities. The process model runs are assigned to different processors for simultaneous execution. The parallel processing capability can take advantage of any number of processors, either on a single computer or linked using a local area network.

This particular parallel processing module uses a dispatcher runner protocol (master – slave) using workstations that are unused or not currently logged into


dispatcher
runner
runner
runner
runner
runner
runner


EXIT

The ucode program uses signal files on the runner processors to let the dispatcher service know its status. Two way communication is required between the dispatcher and runner processes.

File name	Program that writes file	Purpose
jrunner.rdy	JRUNNER	Inform dispatcher that runner is active and ready to make a model run.
jdispatcher.rdy	Dispatcher	Communicate to runner data needed to start model runs.
jdispar.rdy	Dispatcher	Communicate to runner: command to be used to start a model run, parameter values to be used, and that runner should initiate a model run.
jrundep.rdy	JRUNNER	Communicate model-calculated dependent values from runner to dispatcher.
jdispatcher.fin	Dispatcher	Stop execution of runner.
jrunner.fin	JRUNNER	Communicate to dispatcher that signal to stop execution has been received, and that runner is stopping.
jrunfail.fin	JRUNNER	Communicate to dispatcher that runner has encountered an error and is stopping.

The modeler running the model needs to be able to access all the client computers and execute programs on them.

Ideally the runner processes would be started on remote workstations using scripts or a program such as psexec. However, Windows security will not allow the execution of a remote program to have network access.

We initially decided to use a privileged account to allow the user to use remote desktop to log into the remote workstation and start the runner processes.

Copying files between the various workstations could be accomplished with scripts. Other procedures such as remotely killing processes or rebooting could also be accomplished with scripts.

Access to computers was limited by editing the users profile. This limited the use of computers assigned to other users.

Next we decided to try using a user account with normal access. To allow the transfer of files we created shares limited to that user. To allow remote access we created a remote users group, and created an OU populated with the computers we wanted to use, and created a group policy allowing that group remote access.

The limitations with this procedure include creating a lot of shares, and the need for the user to log into the remote workstation to kill processes and reboot.

The runner process takes up 99% of the processor even when it is not processing. This would not be acceptable in an environment where other users need access to a workstation

Normal login	Elevated login
Remote login (remote login group access)	remote login (bases on elevated permissions)
File copy – use shares, scripts to copy files	use the default admin shares as part of path
Must remote login to reboot/kill processes	can use scripts to reboot/kill process
Cannot easily use workstations assigned to an individual user.	Problematic security having elevated access to another users computer

http://igwmc.mines.edu/freeware/ucode/

UCODE_2005 and Six Other Computer Codes for Universal Sensitivity Analysis, Calibration, and Uncertainty Evaluation
Constructed using the JUPITER API
JUPITER: Joint Universal Parameter IdenTification and Evaluation of Reliability
API: Application Programming Interface