Replay of Digitally-Recorded Holograms Using a Computational Grid

J.J. Nebrensky and P.R. Hobson
School of Engineering and Design,
Brunel University, Uxbridge UB8 3PH, UK

Since the calculations are independent, each plane within an in-line digital hologram of a particle field can be reconstructed by a separate computer. We investigate strategies to reproduce a complete sample volume as quickly and efficiently as possible using Grid computing. We used part of the EGEE Grid to reconstruct multiple sets of planes in parallel across a wide-area network, and collated the replayed images on a single Storage Element such that a subsequent particle tracking and analysis code might then be run. Although most of the sample volume is generated up to 20 times faster on a Grid, there are some stragglers which cause the reconstruction rate to slow, and a significant proportion of jobs get lost completely, leaving blocks missing from the sample volume. In the light of these experimental findings we propose some strategies for making Grid computing useful in the field of digital hologram reconstruction and analysis.

This is an expanded version of a paper presented at *OSA Topical Meeting on Digital Holography 2007*, Vancouver, Canada.

OCIS Codes: (090.0090) Holography; (090.1995) Digital Holography; (100.6890) Three-dimensional image processing; (180.6900) Three-dimensional microscopy; (350.4990) Particles

Introduction

In some situations, such as the non-invasive study of marine organisms [1], it is necessary to look at or measure the specific details of the individual particles in a volume, such as their size, shape and relative position. Holography can be used to take a 3-d "snapshot" of the particle field, and the recent substitution of solid-state image sensors for photographic materials allows convenient data capture and storage without the need for chemical processing, as the objects can be reconstructed from a stored digital image (and analysed) by computer.

As the numerical replay step is expensive in terms of both computing power, with multiple 2-d Fourier transforms needed for each depth slice, and intermediate data storage we are investigating the application of Grid computing to this computational challenge. Given the availability of single CCD arrays with nearly 100 million pixels the computational and storage demands of entire volume replay are beginning exceed what is practical even for a powerful desktop machine. Our general approach is to submit the hologram image to a distributed set of worker nodes, each of which computes one or more replayed images each representing a slice across the volume at some depth. These slices are then transferred back to a single storage facility, which thus holds a digital representation of the entire sample volume for further analysis. Recently a number of very large-scale scientific projects have provided some access for other users to their production Grids. Our aim was to investigate whether such a

facility could be of benefit to scientists reconstructing digital holograms. Other approaches to this computational challenge such as using dedicated parallel processing (see for example Ng et al. [2]), while relevant to the problem, are not considered further here. We have used a subset of the EGEE Grid, which currently has over 75 000 CPUs in total spread across around 250 sites around the globe. As minor users of this Grid infrastructure we had no control over its configuration, thus it was not possible to evaluate any explicit optimization scheme that involved the reconfiguring or reprioritizing of resources.

Previously we have found that the overheads in processing and transferring individual depth slices across the Grid result in minimal gains over using a standalone PC for replay [3]. We report here on the efficacy of improved strategies for Grid submission and use, and demonstrate the reconstruction of a sample volume consisting of microscopic particles dispersed in a tank of water recorded with in-line digital holography.

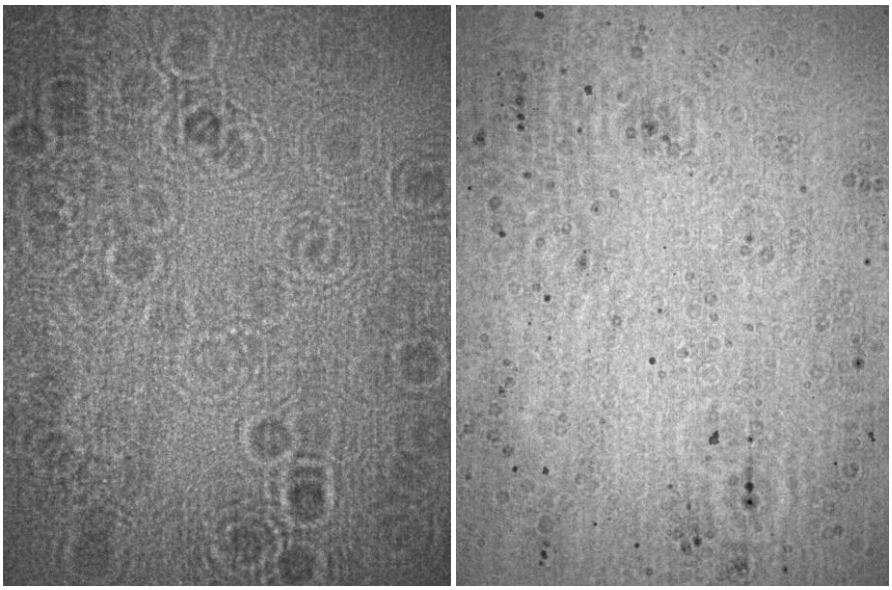

Figure 1 (left): An extract (10 mm wide) from a digital hologram of cenospheres in water.
Figure 2 (right): The region of a numerically-reconstructed arbitrary plane through the sample volume corresponding to figure 1.

1 Digital Holography

We have recorded in-line (Gabor) holograms of objects in water (figure 1) using an 8 megapixel camera (Atmel Camelia 8M, 2300 by 3500 pixels with 12-bit depth) with a collimated beam from a c.w. HeNe laser (λ=633 nm, 1 mW). Our reconstruction software *HoloPlay* reconstructs a single depth plane through the object (figure 2) by de-convolution of the diffraction integral, a process in which the result for any one plane is computationally completely independent of all others (often termed "embarrassingly parallel"), so that the images of many depth planes may readily be calculated at the same time speeding up the replay of the entire volume. Previously we have found that if only a small number of slices (<40) are to be reconstructed then the extra overheads of Grid submission mean it is faster and more reliable to simply replay them sequentially on a single computer [3]. As the Grid is still being developed, it is not yet perfectly reliable and there are also difficulties caused by not all jobs completing and returning results.

For the present work we have recorded a sample volume consisting of cenospheres mostly 100-300 μm dia. (Fillite Trelleborg Specialty Grade (High Alumina) SGHA 500 [4]) dispersed in a water tank. We then numerically reconstructed the water column as a series of slices with 0.1 mm spacing in depth (total 2200 slices = 40 GB when compressed). This spacing was chosen to ensure that there always existed a plane where cenospheres as small as 30 μm dia. would be replayed with good signal-to-noise ratio. Our reconstruction software *HoloPlay* reconstructs single image planes from in-line holograms. It uses the well-known FFTW library (v. 3.0.1) [5] for fast Fourier transform routines, and the same source code compiles and runs both with at least Visual C++ 6 on Windows 2000, and with GCC 3.2 on Linux (Red Hat Linux 7.3 and Scientific Linux 3). Upon its release on an open source basis it has since been re-named "*HoloReco*" [6], but for consistency we refer to it here by its older name.

Upon execution *HoloPlay* reads in control parameters, such as the name of the hologram image file or the wavelength of light, from a simple text file. It is thus possible to direct *HoloPlay*'s operation from a shell script by modifying this control file.

2 Grid Computing

The term "Grid Computing" is commonly found in contexts ranging from world-wide distributed computing systems to traditional parallel processing. We refer here to the paradigm of widely separated, heterogeneous resources proposed by Foster and Kesselman [7, 8] and make use of some of the resources of the EGEE Grid [9].

2.1 How does a Grid job happen?

On a "User Interface" (UI) node, the grid user must specify the executable, data files and other requirements for a particular Grid job using JDL (Job Description Language) (see listing 2 for an example). Upon job submission the UI client passes the input sandbox, containing this JDL and associated files, to a Resource Broker (RB), which identifies the best resource on which to run the job. The Grid currently provides two main classes of resource:

- a Computing Element (CE) provides CPUs
- a Storage Element (SE) provides storage space (disk or tape)

A CE consists of a Gatekeeper (GK), which receives the job, and a set of Worker Nodes (WN) that do the actual calculation – similar to a traditional batch farm. An SE only provides storage space and thus cannot run the job directly, but as WNs will require efficient network access to read or write data, specification of an SE within the JDL may affect the Resource Broker's choice of CE.

After finishing the job the WN returns the output sandbox – containing job output and log files – back to the Resource Broker, from where it can be collected by the user with the UI client.

Rather than having to deal with huge numbers of individuals, Grid resources grant access to "virtual organizations" (VOs), which are dynamic, multi-institutional groups of users with a common problem or application in mind [7, 8]. Resources use X.509 certificates to authenticate individuals as members of a supported VO, so Grid jobs must also include a valid certificate proxy, to confirm the submitter's membership of an appropriate VO.

2.2 How can we use the Grid for Digital Holography?

Reconstruction of any slice is independent of all the others ("embarrassingly parallel"), so we use the Grid to reconstruct many depth planes at the same time. A simple approach is as follows:

- Store digital hologram (and *HoloPlay* binary) on an SE.
- Submit control file for each plane to the Grid. Each job will upload the slice it has reconstructed to a common SE.
- On a WN local to the SE holding the reconstructed slices, run some tracking and identification code to locate objects of interest or perform other data analysis [10].

- Recover results of analysis to UI.

We consider here only the first two steps, corresponding to the re-creation of the sample volume as a series of images stored on the Grid.

Our sample holograms are 2300 by 3500 pixels with 12-bit depth. We use PGM format image files, which can be up to 40 MB in size, so the hologram and images are compressed using *gzip* before being uploaded to the SE (this also allows integrity checking). Typically, a single slice image took up to 2 min. to replay and compress, and around 10 s to upload over a WAN to the SE.

A copy of the reconstruction program, *HoloPlay* and of the hologram image file are placed on a Grid-accessible SE associated with the BITLab facility at Brunel University. A set of job requests is then sent to the Grid, each of which is to download the program and hologram, reconstruct one or more slices across the depth of the sample volume, and upload the resulting images back to the SE. The Grid infrastructure then passes these jobs out to CEs around the globe (see the UML sequence diagram, figure 3).

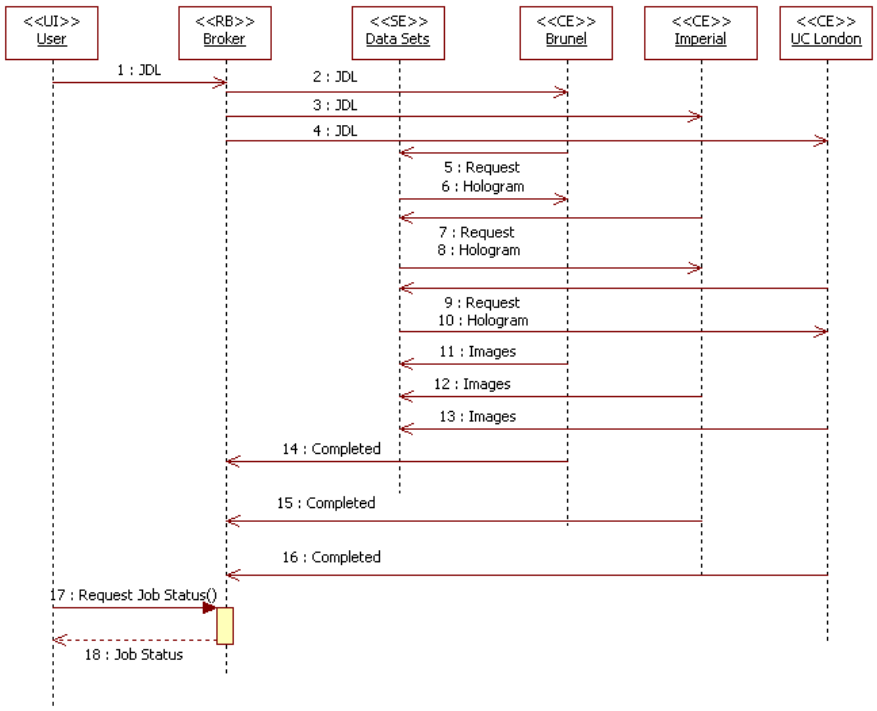

Figure 3: UML sequence diagram representing the replay of digital holograms on the Grid.

In previous work we have found that if only a small number of slices (<40) are to be reconstructed then the extra overheads of Grid submission mean that it is faster and more reliable to simply replay them sequentially on a single computer [2] than to submit them individually to the Grid. As we were submitting relatively small numbers of Grid jobs, we needed little more than the standard EGEE UI tools (*edg-job-submit* for job submission, and the LCG GUI *edg-wl-ui-jobmonitor*, figure 4, for job monitoring).

In the present work we demonstrate the reconstruction of an entire sample volume with 0.1 mm spacing, as a series of 2200 images (a total of 40 GB of compressed data). To reduce overheads we have split the overall task into a smaller number of Grid jobs, each of which replays a block of 10, 50 or 100 slices. It has therefore been necessary to create a set of shell scripts that wrap around *HoloPlay* and *edg-job-submit* in order to create, track, and recover output from the 100's of Grid jobs associated with each volume.

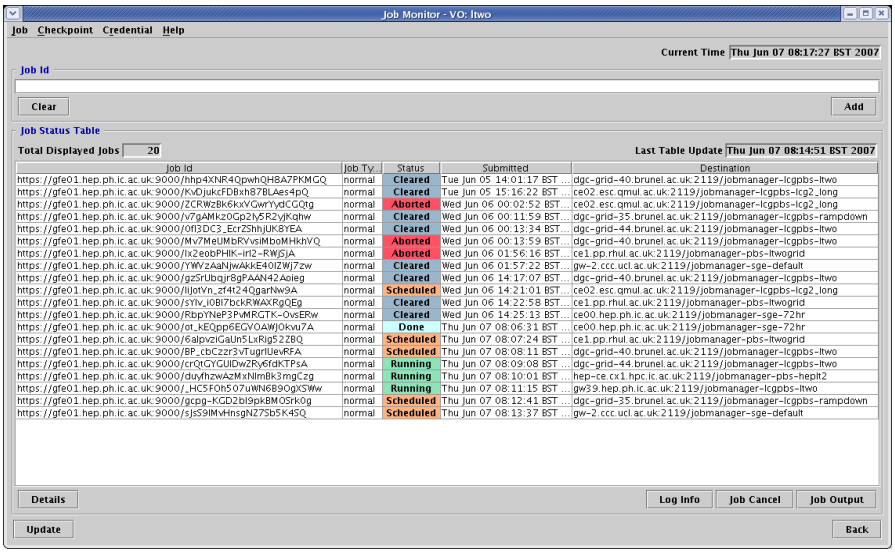

Figure 4: LCG GUI *edg-wl-ui-jobmonitor*: a list of Grid jobs is shown along with their present status and the CE to which they have been sent.

2.3 Digital Holography on the Grid

The *HoloPlay* code was compiled as a single, generic *i*386, statically-linked binary. When run, it reads in a control file *HoloPlay.ini* – which specifies the hologram image to replay, optical parameters to use, etc. – and reconstructs just the one, specified depth plane. As we would like to reconstruct a series of images in turn, we therefore prepared a template for the control file (listing 1) that has fixed parameters including replay wavelength (633 nm), pixel size (11 µm) and replayed image file name

(*Output.PGM*); and unique tokens (*IFILE* and *DEPTH*) for the desired hologram file name and axial position of that slice, respectively.

The *HoloPlay* binary is then invoked from a wrapper script (listing 2) which takes three arguments: the name of the hologram file to use, which slice to start from, and the number of slices to be reconstructed. The first section of the script records some details about the computer for debugging purposes, and then uses *globus-url-copy* to download *HoloPlay* and the requested hologram from an SE (a UML activity diagram is given in figure 5*a*).

The central section of the wrapper is a loop that on each pass uses Linux' *sed* tool to substitute the desired hologram file name and axial position of the current slice into the *HoloPlay.ini* template; runs *HoloPlay* and then compresses and uploads (with *srmcp*) the resulting image file to an SE. For simplicity, we identify the slices using a four-digit number that represents their distance from the hologram in units of 1/10 000 of a metre; thus slice 3456 will be an image of the plane 0.3456 m along the axis from the sensor, and will be stored in the file *Output3456.PGM*.

Finally, the script tidies up by deleting any large files that would otherwise be left behind.

Clearly, by commenting out the data transfers from the script it can be used to generate the entire test volume on the local machine, e.g. by invoking it directly as

```
./holoplay_wrap.sh Fillite11.PGM 2500 4699
```

to reconstruct 2200 slices from hologram *Fillite11.PGM* spanning the depth range containing the water tank at 0.1 mm spacing. When run remotely, these arguments will need to be supplied to the script by the Grid middleware. As well as creating the output image files themselves, the script also displays various messages regarding progress, including the time at which each result is completed (uploaded to SE), from which the progress of the overall task can be understood.

The JDL template used for Grid submission is given in listing 3. It defines the job as being run under the auspices of the LTWO VO, and nominates the wrapper script (listing 2) to be run at the WN. It then specifies the names of the files to which the job's standard input and output should be re-directed, and lists the files that should be sent to the WN in the input sandbox (the wrapper script and *HoloPlay* control file template) and which should be recovered to the UI afterwards (the re-directed standard input and output). Having defined the job itself, the JDL then tells the RB how it should be handled; the Grid should not attempt to repeat jobs it thinks have failed, and a list of the requirements that the CE must fulfill to run the job successfully is given: e.g. the CE must allow the job to run for at least 58 minutes and provide more than 512 MB of memory, and here we also exclude a particular resource where we have encountered problems. At the time this work was done the resources available on the Grid were unusually homogenous; one would normally also specify the target architecture and operating system needed by the compiled binary. It is also possible to define a ranking condition to select among multiple resources that meet the requirements; the default is that the RB will choose the CE that is expected to start running the job first.

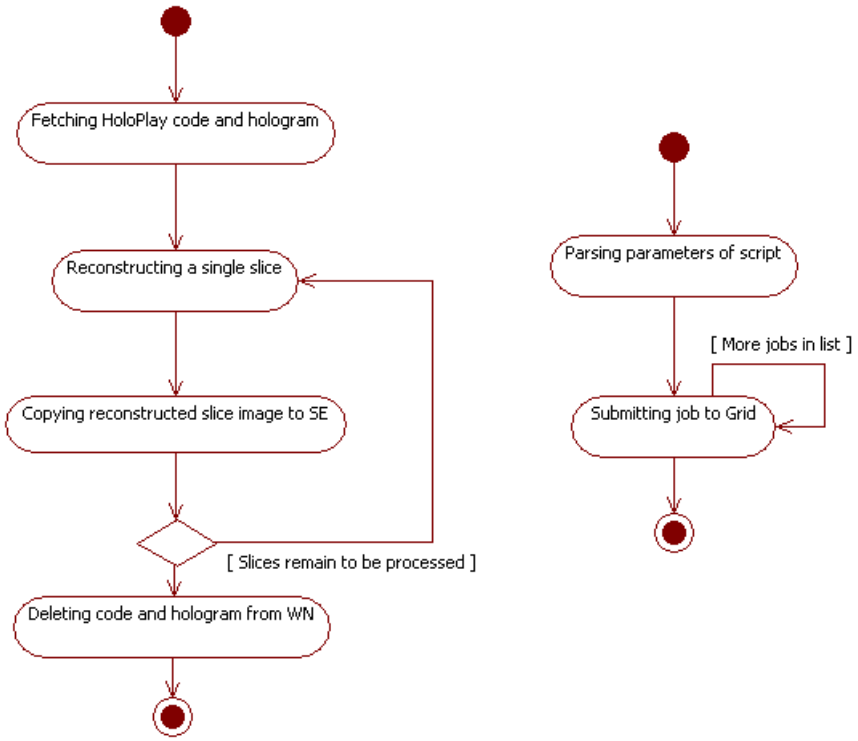

Figure 5*a* (left): UML Activity diagram illustrating the process of running a single *HoloPlay* job on a WN. Figure 5*b* (right): UML Activity diagram illustrating submission of a series of *HoloPlay* jobs to the Grid.

The command-line arguments to be passed ultimately to the wrapper are again tokens, which are substituted by the job submission script (listing 4). This specifies the hologram file and volume to be reconstructed, and the number of slices to be done by each Grid job. It then has a simple loop in which the appropriate values are put into the JDL template and successively submitted to the Grid using the standard *edg-job-submit* command, with housekeeping information being copied to the *joblist* file.

From the user's perspective, one simply has to create a valid Grid proxy, set the granularity (here, 10, 50 or 100 slices per Grid job) in the submission script (listing 4) and then run it. This script then repeatedly fills out the JDL template (listing 3), creating a series of jobs that it then submits to the Grid (a UML activity diagram is given in figure 5*b*).

The Grid RB examines each job in turn and sends it to the resource that is currently the best match to the requirements, where it will wait its turn in the queue. As resources are shared among many VOs and users with different priorities, this can be

unpredictable and no assumption should be made that the jobs will be run in the order that they were submitted.

When a Grid job does reach a WN, the input sandbox is unpacked and the wrapper script invoked with the parameters passed along from the submission script. This then fetches the *HoloPlay* binary and hologram, and cycles through reconstructing each of its allotted slices and uploading it to the SE. After the wrapper finishes, the output sandbox will be sent back to the to the RB.

Further scripts, not included here, use the *joblist* file to track the Grid jobs and download the output sandboxes to the UI, and then extract from them the completion time (after the replayed image has been successfully uploaded to the SE) of every slice in the sample volume. Grid jobs that did not run as expected were monitored manually using the LCG GUI *edg-wl-ui-jobmonitor* tool, figure 4, which allows convenient browsing of job status and logging information.

To understand the performance of the Grid for digital holography, on a number of occasions during 2007 we submitted batches of Grid jobs each reconstructing between 10 and 100 single slices and measured how long it takes between starting the job submission and the replayed images arriving back at the SE. The Grid deployment was done within the LTWO VO giving access to resources within the London Tier 2 of the UK GridPP project [11], which forms part of the EGEE Grid. LT2 is a collaboration which had a total of ~3000 CPUs across 7 institutes spread across London, UK, at the time this work was done. Although their primary role is the analysis of High-Energy Physics data, the resources are also available to internal users from the participating institutes, although the LTWO VO may have lower priority or a limited share, so the number of Grid jobs that we would expect to have running at any one time is significantly smaller. For convenience we uploaded the replayed images to the same SE that held the binary and hologram; we used the LCG DPM [12] installation at Brunel University consisting of a head node and at least 3 separate RAID-5 pool nodes to reduce disk access clashes between simultaneous data transfers. At the time all, but one, sites were interconnected with 1 Gbps or better WAN links. Note that at the time this work was done EGEE production services were still based on the EDG middleware rather than gLite.

Misconfigured sites and middleware problems meant that Grid job efficiency was about 90% – i.e. 10% of jobs failed completely (never returned data). The Grid infrastructure can resubmit them internally, but this can take over 12 hours and, as there are a number of failure modes not detected by the Resource Broker, this feature has been turned off here.

3 Results

It can be seen from figure 6 that after submission starts there is usually an initial delay before the first results are returned (while the jobs pass through the Resource Broker and CE batch queues) followed by a rush of results covering over half the total volume. However, once about 2/3 of the results have been done, the rate starts to tail off, as the remaining jobs are those stuck in long queues, on unusually slow machines or suffering some other problem.

Table 1 summarises the results of a number of submissions of *HoloPlay* to the LT2 Grid resources. In the table the elapsed time, normalized to a single dedicated local processor (desktop PC), to process 50%, 70% and 90% of the 2200 hologram depth slices is given. The 100% column gives the normalized time for the complete data set to be produced.

From table 1 a number of conclusions can be drawn. Firstly for the 50th and 70th percentile there is a very significant performance gain compared to a single PC. Waiting for 90% of the data set to be produced shows the influence of the small fraction of jobs that get into batch queues that are already occupied. Finally it should be noted that only 2 out of 9 attempts resulted in the complete data set being returned. This is the effect of the ~90% reliability of the Grid – if one Grid job in ten fails, then successful replay of a volume requiring more than twenty such jobs seems unlikely (indeed, run "10slices 1", which eventually recreated 2190 slices from 220 Grid jobs, is actually a notable success!). While the reliability of the Grid middleware itself has been improving over the years, it is still far from perfect and there are also external factors, such as job queueing times being so long that the X.509 proxy certificates had expired before the job could run and problems with WNs being rebooted or failing during job execution (no job checkpointing was used). The exact cause cannot always be identified remotely. In the work reported here we have not encountered any failed data transfers to or from the SE; the failures can all be ascribed to problems either with the RB or the batch queuing system at the CE (see table 2).

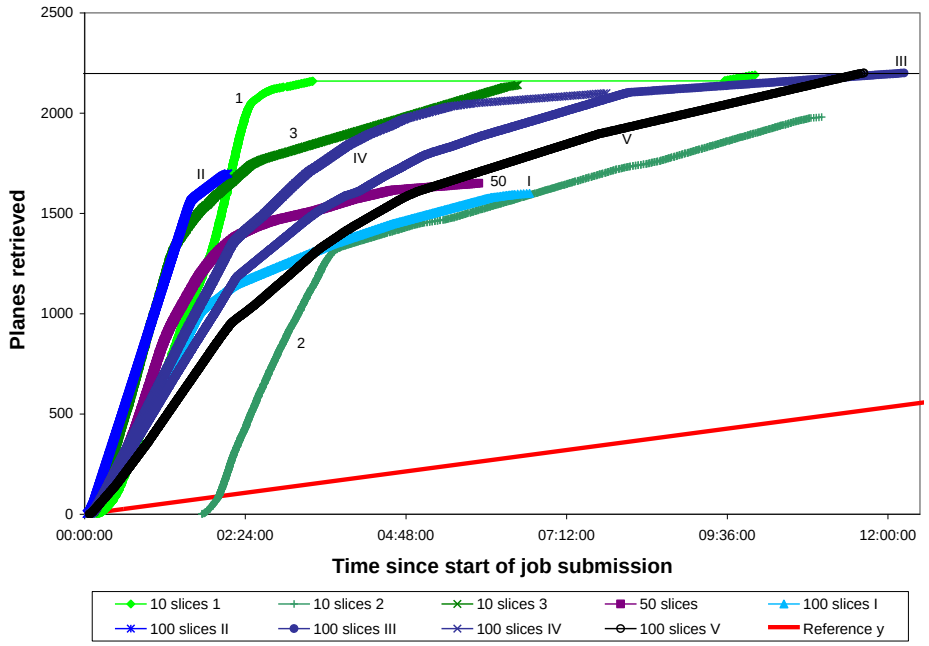

Figure 6: Overall rate of job completion (times are shown in hh:mm:ss format). The Roman and Arabic numerals denote different submissions to the Grid.

Proportion of slices replayed:	50% (1100)	70% (1540)	90% (1980)	100% (2200)
10 slices 1	15.0	16.7	18.6	n/a
10 slices 2	7.4	5.7	4.0	n/a
10 slices 3	22.0	19.1	9.3	n/a
50 slices	16.0	9.3	n/a	n/a
100 slices I	12.0	6.2	n/a	n/a
100 slices II	21.8	22.3	n/a	n/a
100 slices III	11.7	9.5	6.5	4.0
100 slices IV	13.6	12.3	9.1	n/a
100 slices V	9.0	7.6	5.1	4.3

Table 1: Relative rate with which the given percentage of reconstructed images were uploaded to the SE, compared with serial replay on a single processor (AMD Athlon XP model 10 2600+ (1920 MHz), 1.5 GB memory).

Run ID:	Date:	Grid jobs submitted:	Grid jobs failed:
10 slices 1	11 June 2007	220	1% at CEs
10 slices 2	30 August 2007	220	10% at CEs, 0.5% at RB
10 slices 3	4 September 2007	220	3% at CEs
50 slices	19 June 2007	44	25% at CEs
100 slices I	12 June 2007	22	27% at CEs
100 slices II	11 June 2007	22	23%, unknown
100 slices III	20 June 2007	22	No Failures
100 slices IV	3 September 2007	22	5% at CEs
100 slices V	3 September 2007	22	No Failures

Table 2: Rate of failure of Grid jobs

Conclusions

We have created a numerical replay code for digital holography that has been run on a major production Grid, and we have demonstrated the reconstruction and storage of a large sample volume using this Grid. We have achieved replay rates from jobs distributed around a wide-area network of around ten times that of a single desktop computer, without needing any additional local investment in computer hardware or services (electricity and cooling). Our results will apply to any task with similar computing requirements and data access patterns and are not specific to the algorithms used here – e.g. instead of megapixel holograms, one could envisage each Grid job reconstructing the whole sample volume from one CCIR frame, with the aim of building up a complete time series (4-d analysis).

Digital hologram reconstruction, although "embarrassingly parallel" has some important differences in practice from the production of Monte Carlo samples for HEP (to date, the largest user of the EGEE Grid). The reason for this is that the final result consists of highly correlated images that are depth slices of a whole recorded volume, rather than a collection of independent data sets. We suggest that the following strategies should be used when using the Grid computing paradigm for volume reconstruction:

1. **Visualisation of an entire volume to locate regions of interest:** Here losing a few slices is probably not important compared to being able to rapidly view the volume in its entirety, so one could submit jobs that sample the whole volume, with the slices from one job interleaved with those from the next. A lost Grid job would then give poorer depth resolution, rather than leaving a whole chunk missing. In this mode the significant gain that the Grid provides over a single processor, at least to reconstruct 50% of all the volume slices is a significant advantage. An important complication here is ensuring that the interleaved jobs are indeed submitted to different nodes or sites, rather than sent by the RB to the same (faulty) resource.

2. **Reconstruction coupled with image processing** (to select particular types of objects in replayed images for example): Here it is essential to have images closely sampled in depth available on the same processor. As it is not known *a priori* which slice will contain the in-focus image some overlap between volume samples is essential. For efficiency this suggests a relatively small sample of jobs each reconstruction, of order 100 sequential slices. The strategy here should be to understand after what time (70^{th} or 90^{th} percentile) we enter the regime where the final slices will take an inordinately long time to return to the SE (or will in fact never be completed). These could then be re-submitted pre-emptively.

The Grid middleware itself is in a state of continuous development. Two features that we hope to explore in further work are the submission of parameterised jobs via the new "gLite WMS" RB, and the use of the LFC file catalogue [12] and utilities to simplify output file storage at the remote sites.

Acknowledgements

The authors wish to thank Marc Fournier-Carrié and Paul Fryer for their work in developing the replay software, and Trelleborg Fillite Ltd. for providing the cenosphere sample. Presentation of this work at the OSA Topical Meeting on Digital Holography 2007, Vancouver, Canada was made possible by grant ITG E7-489 from The Royal Academy of Engineering, UK.

References

1. P.R. Hobson and J. Watson: "The Principles and Practice of Holographic Recording of Plankton" *Journal of Optics A: Pure and Applied Optics* **4**, pp. S34-S49 (2002)

2. T.W. Ng, K.T. Ang and G. Argentini: "Temporal Fringe Pattern Analysis with Parallel Computing" *Applied Optics* **44** (33) pp. 7125-7129 (2005)

3. J.J. Nebrensky and P.R. Hobson: "The Reconstruction of Digital Holograms on a Computational Grid" in *Holography 2005: International Conference on Holography, Optical Recording, and Processing of Information - Proceedings of SPIE*, **6252** Art. CID: 62521I (2006)

4. Trelleborg Fillite Ltd., Goddard Road, Astmoor Industrial Estate, Runcorn, Cheshire, WA7 1QF, UK

5. The FFTW Project home page, http://www.fftw.org/

6. The *HoloReco* home page, http://sourceforge.net/projects/holoreco/

7. I. Foster, C. Kesselman and S. Tuecke: "The Anatomy of the Grid: Enabling Scalable Virtual Organizations" *International Journal of High Performance Computing Applications* **15**, pp. 200-222 (2001)

8. I. Foster and C. Kesselman: "*The Grid: Blueprint for a New Computing Infrastructure*" 2nd revised Ed. Morgan Kaufman (2003)

9. The EGEE project home page, http://www.eu-egee.org/

10. J.J. Nebrensky, P.R. Hobson and P.C. Fryer: "Grid computing for the numerical reconstruction of digital holograms" in *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments III*; Ryszard S. Romaniuk, ed. *Proc. SPIE* **5775**, pp. 285-296 (2005)

11. The GridPP collaboration: "GridPP: Development of the UK Computing Grid for Particle Physics" *Journal of Physics G: Nuclear and Particle Physics* **32**, pp. N1-N20 (2006) or see http://www.gridpp.ac.uk/

12. "Official Documentation for LFC and DPM" https://twiki.cern.ch/twiki/bin/view/LCG/DataManagementDocumentation

Listing 1: Control file template for *HoloPlay*. Before it runs, the actual values are substituted for the tokens IFILE and DEPTH by the wrapper script, listing 2.

```
1  [Parameters]
2  InputFile= IFILE
3  OutputFile= Output.PGM
4  Wavelength= 633e-9
5  Pixel_size_X= 11.4e-6
6  Pixel_size_Y= 11.4e-6
7
8  [Reconstruction Parameters]
9  Distance_from_hologram= DEPTH
10 Apply_Zero_Padding = Yes
11 Planner_Vigour= FFTW_MEASURE
12
13 [Debug]
14 Write_Zero_padded_image_to_file= No
```

Listing 2: Wrapper script for a typical *HoloPlay* job.

```
#!/bin/sh
#usage: holoplay_wrap.sh <hologram> <startslice> <numslices>
# Logging info
echo "Running on `hostname -f`"
echo "Processor - `cat /proc/cpuinfo |grep 'model name'`"
echo "Physical memory - `cat /proc/meminfo | grep 'MemTotal'`"

HOLO=$1
# Fetch hologram and code
echo "Fetching data at `date`"
globus-url-copy gsiftp://dgc-grid-34.brunel.ac.uk/storage/for/LCG/brnbl/holoplay/${HOLO}.gz file://`pwd`/${HOLO}.gz
globus-url-copy gsiftp://dgc-grid-34.brunel.ac.uk/storage/for/LCG/brnbl/holoplay/holoplay386.gz file://`pwd`/holoplay.gz
echo "Data received at `date`"
gunzip ${HOLO}.gz
gunzip holoplay.gz; chmod +x holoplay


CURR_ID=$2
SLICES=`expr $3 - 1`
LAST_ID=`expr ${CURR_ID} + ${SLICES}`
# Increase by one less than no. of slices. Increase by 0 for just one slice

echo "Reconstructing slices ${CURR_ID} to ${LAST_ID}"

until [ ${CURR_ID} -gt ${LAST_ID} ]; do

  sed s/DEPTH/0`echo "scale = 4; ${CURR_ID} / 10000;" | bc`/ HoloPlay.ini.in | sed s/IFILE/${HOLO}/ > HoloPlay.ini
  ./holoplay
  ls -l

  mv Output.PGM Output${CURR_ID}.PGM
  gzip -9 Output${CURR_ID}.PGM
  echo "Storing result as Output${CURR_ID}.PGM at `date`"
  srmcp \
    file:///`pwd`/Output${CURR_ID}.PGM.gz \
    srm://dgc-grid-34.brunel.ac.uk:8443/srm/managerv1?SFN=/dpm/brunel.ac.uk/home/ltwo/digiholo/test01/Output${CURR_ID}.PGM.gz
  echo "Result stored at `date`"

  rm Output${CURR_ID}.PGM.gz

  CURR_ID=`expr ${CURR_ID} + 1`
done


# Tidy up
rm ${HOLO}
rm holoplay
if [ -e Temp_zero_padded.pgm ] ; then
  rm Temp_zero_padded.pgm
fi
ls
```

Listing 3: JDL template file for a typical *HoloPlay* job

```
[
VirtualOrganisation = "ltwo";
Executable = "holoplay_wrap.sh";
Arguments = "arg1 arg2 arg3";
StdOutput = "holoplay.out.txt";
StdError = "holoplay.err.txt";
InputSandbox = {
   "holoplay_wrap.sh",
   "HoloPlay.ini.in"
};
OutputSandbox = {
   "holoplay.out.txt",
   "holoplay.err.txt"
};
RetryCount = 0;
requirements = IsMember("GRIDPP-LT2", other.GlueHostApplicationSoftwareRunTimeEnvironment)
   && (other.GlueCEPolicyMaxWallClockTime>58)
   && (other.GlueCEInfoHostName!="mars-ce2.mars.lesc.doc.ic.ac.uk")
   && (other.GlueHostMainMemoryRAMSize>512 );
JobType = "normal";
Type = "Job"
]
```

Listing 4: Job submission script to deploy *HoloPlay* on to Grid.

```
1  #!/bin/sh
2  HOLOFILE=Fillite11.pgm
3  CURR_ID=2500
4  SLICES=100
5  LAST_ID=4700
6
7  date >> joblist
8
9  until [ ${CURR_ID} -gt ${LAST_ID} ]
10 do
11       sed s/arg1/${HOLOFILE}/ holoplay.jdl.in | sed s/arg3/${SLICES}/ | sed s/arg2/${CURR_ID}/ > holoplay.jdl
12       echo -n "${CURR_ID} " >> joblist
13       edg-job-submit --vo ltwo --nomsg holoplay.jdl | grep http >> joblist
14       sleep 10s
15
16       CURR_ID=`expr ${CURR_ID} + ${SLICES}`
17 done
18
19 date >> joblist
```