Tracking PACS Usage with Open Source Tools

Todd L. French · Steve G. Langer

Published online: 10 September 2010
© Society for Imaging Informatics in Medicine 2010

Abstract A typical choice faced by Picture Archiving and Communication System (PACS) administrators is deciding how many PACS workstations are needed and where they should be sited. Oftentimes, the social consequences of having too few are severe enough to encourage oversupply and underutilization. This is costly, at best in terms of hardware and electricity, and at worst (depending on the PACS licensing and support model) in capital costs and maintenance fees. The PACS administrator needs tools to assess accurately the use to which her fleet is being subjected, and thus make informed choices before buying more workstations. Lacking a vended solution for this challenge, we developed our own.

Keywords Clinical use determination · Computer systems · Cost savings · Data mining

Background

Among radiology informatics software (Radiology Information System (RIS), Picture Archiving and Communication System (PACS), Speech Recognition) vendors, there are numerous licensing schemes: sometimes, a vendor may charge a per seat fee for their software, others may charge for the number of simultaneous connections to a server, and still others by the volume of a site's exams. Even if a given site may pay by volume, it is fiscally irresponsible to have

more workstations than are truly needed; the concomitant hardware, display, and management costs cannot be tolerated in the shrinking reimbursement environment. For instance, the hardware costs alone for a diagnostic PACS station at our site is about \$23,000 (workstation \$5K, RIS display \$200, speech recognition microphone and accessories \$500, four diagnostic displays \$17,200).

Our site, like many large institutions, has over 200 PACS-related workstations. The question naturally arises, "Are all of them required?" It is easy to know when a required workstation is not present; users are quite accomplished at communicating when such a shortfall exists. The opposite is not obvious. A surplus workstation simply sits in the corner unused, burning electricity, hardware, support costs—and money. As a cost containment measure, we were tasked with identifying underutilized PACS workstations. As an organization, we hypothesized that many workstations were not being fully used but had no method to identify these systems. To assure that our site has only the workstations it truly needs, we first attempted a simple survey with the staff. A clipboard was first tried near each workstation, and we asked for radiologist's initials and date to count users. Compliance was low. We then asked for date only, but compliance was not much better. An oral survey was then tried to query radiologists as to which workstations were most and least used. Recollections of the most used workstations were quite accurate, but the least used was widely disparately reported based on who answered the question. We ultimately decided an automated tool was needed and tried to work with our PACS vendor to implement a monitoring policy, but surprisingly, we were informed such tools did not exist. We then looked at the literature for an applicable tool. There are some good options, but they were either overkill or not really

T. L. French · S. G. Langer (✉)
Mayo Clinic,
Rochester, MN, USA
e-mail: langer.steve@mayo.edu

T. L. French
e-mail: french.todd@mayo.edu

targeted to the task at hand [1, 2]. We therefore set about to create our own solution.

Ideally, the PACS vendor would provide administration software to track usage, but at our site, there was no software available to do the task required. Attempting to answer this question for the PACS fleet led us to a Free Open Source Software (FOSS) solution. As it turned out, the FOSS alternative to commercial software was probably easier to set up and use than the commercial alternative would have been. This presentation will detail how the task was accomplished.

Methods

A design goal for the project was to have a simple Web site that administrators could reach that would accurately tally the usage (defined by studies loaded per workstation) over a given time period. Fortunately, the FOSS world is blessed with many tools that are tailored for exactly this task (web reporting of results in a database). In fact, there is even an acronym for the particular assemblage of software tools that accomplishes this task—LAMP—(Linux, <http://www.linux.org/>; Apache, <http://www.apache.org/>; MySQL, <http://www.mysql.org/>).

Fig. 1 A listing of the PHP code: it opens a connection to the MySQL database, scans the PACS workstation fleet for study usage data (only counting studies actually opened), and stores the data to MySQL

```
<?php
//connect to database
$con = mysql_connect("foss-1","tlf02","TF901may");
if (!$con)
{
    die('Could not connect: ' . mysql_error());
    echo "bad connector";
}
mysql_select_db("pacs", $con);
//get search strings
$searchstrings = mysql_query("SELECT * FROM searchstrings");
//setup search string array
$i=0;
$searcharray;
while($row = mysql_fetch_array($searchstrings))
{
    $searcharray[$i] = $row['string'];
    //echo $i;
    //echo $searcharray[$i];
    $i=$i+1;
}

//get yesterday's date
$today = date("Y.m.d", mktime(0, 0, 0, date("m") , date("d")-1 , date("Y")));
//get pacnames and other info
$pacinfo = mysql_query("SELECT * FROM pacinfo");
while($row = mysql_fetch_array($pacinfo))
{//start of while loop
    //get values from pacinfo table
    $file = $row['path'];
    $modality = $row['modality'];
    $type = $row['type'];
    $pacname = $row['pacname'];

    $mountstring = "sudo mount_smbfs -N -l " . $pacname . " //root@" . $pacname . "/d$ " .
    "/usr/pacstmp";
    $unmount = "umount -f /usr/pacstmp";
    chdir("/etc");
    //echo $file;
    echo $mountstring . "\r\n\r\n";
    exec($mountstring, $err);

    if (file_exists($file)) //check path for file to parse
    {
        //start of if 1
        $lines = file($file); //read file into memory
        foreach ($lines as $line_num => $line) //read line by line
        {
            //start of for each loop
            //look for search strings
            $pos = ((strpos(" . htmlspecialchars($line) . ", "Remote study
            inserted")) or (strpos(" . htmlspecialchars($line) . ", "Remote study copied")));
            if ($pos !== false)
            {

```

Fig. 1 (continued)

```

$enddate = explode(" ", $line); //parse the line
//echo $enddate[0];
//echo $enddate[6] . $enddate[7] . $enddate[8];

if ($enddate[0] == $today) //check the date located at 0 position

in string
    { //start of if 2
        $name = $enddate[6] . $enddate[7] . $enddate[8]; //grab
the patient name from string last 3 words
        $searchforit = $enddate[0] . $name; // add the date
to the name
        // lets add the number of occurrences
        if ($pos != false)
        { //start of if 3
            If ($searchforit <> $lastvalue) //check to make
sure name changes
                { //start of if 4
                    $count = $count + 1; //add number of
instances
                } //end of if 4
                $lastvalue = $searchforit; //change the name to
the current value for next search
            } //end of if 3
        } //end of if 2
    } //end of for each loop
} //end of if 1
//insert data into table

mysql_query("INSERT INTO pacsdata (`modality`, `type`, `name`, `qdate`, `studies`)
VALUES ('$modality', '$type', '$pacname', '$today', '$count')");
$count = 0;
exec($unmount, $err);

}
mysql_close($con);
?>

```

www.mysql.com/?bydis_dis_index=1 and PHP, <http://www.php.net>) [3].

In our case, we deviated slightly from the pure LAMP solution. One author has more experience with FreeBSD; hence, we could say we had a “FAMP” solution (FreeBSD, <http://www.freebsd.org/>). Also, while the charts could have been rendered purely with PHP libraries, ChartDirector was found to have an elegant user interface (ChartDirector, Advanced Software Engineering, Hong Kong). Of this assemblage, the last item—ChartDirector—was not totally free but had a free 30-day trial. Upon using it, we decided the \$99 price was worth the saved effort that would have been needed to do all charting in PHP. A totally FOSS stack could have used Gnuplot (<http://www.gnuplot.info>), but this tool kit was deemed to have a steeper learning curve [4].

FreeBSD was installed first as a virtual machine (VM) on one of our virtual servers [5]. After the FreeBSD VM was stabilized, all the other components were added. FreeBSD has an extensive software collection available via “ports” that include Apache, PHP, MySQL, and ChartDirector (the free time-limited version); all available on the FreeBSD ports repository. The FreeBSD package manager resolved all interdependencies.

Once all software was installed, a functional web server capable of displaying numerical data in graphical form was available. The project’s required application logic was coded using PHP; PHP has vast amounts of real world examples easily retrievable via the internet [6]. Our main PACS has a simple hub and spoke topology that uses on-demand pull; hence, studies are not pulled to a workstation unless a user actually requests and views them. At first, data acquisition code was developed to:

- attach to each workstation via Common Internet File System,
- open the PACS log file that contains study load information,
- read through the file and capture daily study load data, and
- store data in the MySQL database.

Second, a display script was written to accomplish:

- connecting a web page to the database,
- accepting user requests for date ranges and subsets of the fleet,
- performing a query on the database for the above parameters, and

Fig. 2 A partial listing of the MR PACS workstations at our site. Near each workstation name (i.e., pac00085), there are up to four bars corresponding to the number of studies actually opened during each week of the month period; the bottom bar corresponds to the week of 2010-01-17, the next one up is 2010-01-24, etc. The shading discontinuity in each bar represents the median value for that bar

(d) using ChartDirector and PHP software to render the graphs.

Figure 1 shows the data acquisition code. The display code is available on request (it is too long to include here).

Fig. 3 A Year to Date view of a subset of our PACS workstations during the third quarter of 2009. As in Fig. 2, the shading discontinuity in each bar represents the median value for that bar

Results

Results proved to be better than expected. Once the hurdle of identifying a reliable usage indicator in the PACS log file was overcome, data flowed seamlessly into the MySQL database. Subsequently, the web front end to ChartDirector

easily converted the stored data into graphs for display. The department administrators were easily able to interpret information to help guide their decision-making processes; in all, 18 systems were identified as underutilized, turned off, and now act as spares for the rest of the fleet. Costs were saved by removing those workstations from support contracts. The web query form permitted users to review data based on several time scales including year to date and during an arbitrary 4-week window. Figure 2 shows an example screen that aggregates study-open counts per workstation for each of the prior 4 weeks.

In the next figure, the user has selected a Year to Date view that shows all the studies opened per workstation in the fourth quarter of 2009 (Fig. 3).

Discussion

While open source software may not be a viable solution for every need, there are times when it can be used, and used very effectively. In the scenario outlined herein, a comparable effort could have been made using Windows Server 2003, Internet Information Server (IIS), SQL Server, and Visual Studio (Microsoft Corporation, Redmond, WA, USA). The costs for this solution would be approximately:

- (a) Windows Server 2003 with IIS (\$1,200),
- (b) SQL Server (\$2,000), and
- (c) Visual Studio (\$750).

Fortunately, it seemed that most charting programs (including ChartDirector) were oriented toward PHP rather than Microsoft products. Hence, the FOSS solution proved easier than using commercial licensed software. Another advantage of the project is that if the tool is only needed for a short time, the software can be discarded at no capital cost to the users. The only financial involvement needed is the development time. Also, since our implementation is a virtual machine, it is easy to leave the tool in an inactive state until a new need arises—without having to worry about licensure issues.

Sadly, many major PACS offerings come with little opportunity for the end users to customize configurations and reporting mechanisms. While that suffices for a majority of the functions, there may be administrative needs that are outside the scope of PACS vendors. In this case, a simple usage report was needed. Not only did this effort achieve the expected results, it did so with a minimal amount of out-of-pocket expense. Management obtained their usage reports, and Information Technology (IT) gained valuable experience using nontraditional methods to achieve a traditional outcome. Another advantage of

FOSS components is that people share their experiences and code for the benefit of all via user groups.

Conclusions

PACS are complex entities, with many pieces of hardware and software merged together to form a complete system. Often, when PACS workstations are deployed, managers use a best-guess method to identify potential locations for the workstations, and more workstations than are actually necessary may be deployed. To further complicate the matter, many vendors do not have the necessary tools available to help determine which machines are being used. Because systems may be licensed by workstation and counted under support contracts, it becomes extremely important to ensure all systems are being fully utilized.

On face value, the administrative request to locate underutilized workstations appeared quite simple, but in practice, answering the question with either human-gathered information or commercial tools proved ineffective or expensive. In the present economy, the same managers asking the above question were loath to spend the commercial dollars required to answer it. This left few choices for IT support staff using traditional licensed software. Faced with the problem of determining PACS workstations usage and giving managers near real-time information, our group was driven to a simple yet effective FOSS solution.

Acknowledgments The authors wish to thank the first author's son, Andrew, for artistic assistance on the figures.

References

1. Nagy PG, Daly M, Warnock M, Ehlers KC, Rehm J: PACSPulse: a web-based DICOM network traffic monitor and analysis tool. *Radiographics* 23(3):795–801, 2003. ISSN: 0271-5333; PMID: 12740478
2. Toland C, Meenan C, Warnock M, Nagy P: Proactively monitoring departmental clinical IT systems with an open source availability system. *J Digit Imaging* 20(Suppl 1):119–124, 2007. ISSN: 0897-1889; PMID: 17882485
3. Lee J, Ware B: *Open Source Web Development with LAMP: Using Linux, Apache, MySQL, Perl, and PHP*. Addison Wesley, Boston, 2002. ISBN 0-201-77061-X
4. Janert PK: *Gnuplot in Action: Understanding Data with Graphs*. Manning Publications, Greenwich, 2003. ISBN 1933988398
5. Langer S, Charboneau N, French T: DCMTB: a virtual appliance DICOM toolbox. *J Digit Imaging* 2009 Aug 25. [Epub ahead of print] PMID:19705204. doi:10.1007/s10278-009-9230-8
6. PHP examples. http://www.php-scripts.com/php_diary/php_scripts.html. Last viewed June 2010