Tim Cornwell Consulting

Interferometry and Imaging

Distribution of the Rau-Cornwell MSMFS algorithm

Name	Designation	Affilitation
T.J. Cornwell	Author	TCC
Signature & Date:		

Version	Date of issue	Prepared by	Comments
0.9	30/04/2018	T.J. Cornwell	Initial version
1.0	18/06/2018	T.J. Cornwell	Corrections from Mitch

ORGANISATION DETAILS

Name	Tim Cornwell Consulting

Table of Contents

Document no.: TCC-CASS-001   Unrestricted
Revision: 0.9   Author: T.J. Cornwell
Release date: Monday 18th June, 2018 09:51   Page 3 of 38

List of abbreviations

ASKAP   Australian Square Kilometer Array Pathfinder

ARL   Algorithm Reference Library

FFT   Fast Fourier Transform

FoV   Field of View

MS-MFS   Multi-Scale Multi-Frequency Synthesis

MAM   Multi-Scale Multi-Frequency Synthesis

PSF   Point Spread Function

SDP   Scientific Data Processor

SKA   Square Kilometre Array

List of Figures

Document no.: TCC-CASS-001   Unrestricted
Revision: 0.9   Author: T.J. Cornwell
Release date: Monday 18th June, 2018 09:51   Page 5 of 38

Document no.: TCC-CASS-001   Unrestricted
Revision: 0.9   Author: T.J. Cornwell
Release date: Monday 18th June, 2018 09:51   Page 6 of 38

List of Tables

Document no.: TCC-CASS-001
Revision: 0.9
Release date: Monday 18th June, 2018 09:51

Unrestricted
Author: T.J. Cornwell
Page 7 of 38

Summary

The MSMFS algorithm is integral to the data processing for ASKAP and SKA. The inner loops for ASKAP continuum processing use the Multi-Scale Multi-Frequency Synthesis (MSMFS, also M&M[1]) to model the changing structure of the source over frequency. By synthesising across frequency, the imaging performance of the telescope can be improved.

M&M consists of a major cycle that compares the current model with the visibility data, and a minor cycle that deconvolves in (RA, Dec, scale, moment) space. The major cycle is composed to gridding, degridding, and FFT steps and has seen the most effort in optimisation. The minor cycle of M&M consists of a modified and extended CLEAN in image space. This is currently executed in a single thread and can constitute a significant overhead in that processing is blocked until this CLEAN finishes.

The amount of processing in typical use of M&M is comparable to that in the invert and predict steps. Hence since the M&M step is currently serial, it will inevitably result in a bottleneck. The cure is to reduce the amount of work in M&M and and/or distribute the work across multiple workers.

We analyze the various ways that the distribution of processing can be performed and what performance we can expect. We develop and demonstrate an algorithm. Out analysis shows that strong scaling will possible up to hundreds of nodes.

The algorithm has been coded in python using a graph-based distribution package, Dask. We describe the steps necessary to implement the algorithm in ASKAPsoft using MPI to achieve the distribution.

---

[1]SDP adopted this name since it is easier to say.

Reference Documents

The following documents are referenced in this document. In the event of conflict between the contents of the referenced documents and this document, *this document* shall take precedence.

Reference Number	Reference
[RD01]	Parameterized deconvolution for wide-band radio synthesis imaging, Urvashi Rao Venkata, 2010, PhD thesis
[RD02]	U.Rau, T.J.Cornwell, A multi-scale multi-frequency deconvolution algorithm for synthesis imaging in radio-interferometry, , Astronomy and Astrophysics, Volume 532, August 2011.
[RD03]	Cornwell, T.J. (2016), Implementation of the Rau-Cornwell MSMFS algorithm, TCC-SDP-151123-2.

1 Purpose of the document

The purpose of this document is to describe options for distributing the minor cycle of the Multi-Scale MultiFrequency Synthesis algorithm, as implemented in ASKAPsoft.

2 Scope of the document

The scope includes only the published Multi-Scale, Multi-Frequency algorithm, existing implementations, and inclusion in the ASKAPsoft processing pipelines. Only distribution of the minor cycle of that algorithm is addressed. In addition, the many possible threading possibilities are not explored.

3   M&M algorithm in ASKAPsoft

The M&M algorithm implementation has been discussed extensively in RD03. We use the same terminology and symbols.

The (serial) minor cycle from the Urvashi-Cornwell algorithm [RD01, RD02, RDO3] is shown below in both pseudo-code (Algorithm 1) and python (Figure 1).

Algorithm 1: Minor cycle of M&M

Data: input : number of Taylor-terms N_{Taylor}, number of scales N_{scales}
Data: input : image noise threshold, σ_{thr}, loop gain g
Data: input : scale basis functions : $\vec{I}_s^{\text{shp}} \ \forall s \in \{0, N_{\text{scales}} - 1\}$
Result: model coefficient images : $\vec{I}_t^{\text{m}} \ \forall t \in \{0, N_{\text{Taylor}} - 1\}$

1 **repeat**
2   **for** $s \in \{0, N_{\text{scales}}\text{-}1\}$ **do**
3   Construct $\vec{I}_s^{\text{rhs}}$, an $N_{\text{Taylor}} \times 1$ vector from $\vec{I}_{s_t}^{\text{res}} \ \forall t \in \{0, N_{\text{Taylor}}\text{-}1\}$
4   Compute principal solution $\vec{I}_s^{\text{sol}} = [\underset{\sim}{H}_s^{\text{peak}-1}]\vec{I}_s^{\text{rhs}}$
5   Fill solution $\vec{I}_s^{\text{sol}}$ into model images $\forall t : \vec{I}_{s_t}^{\text{m,sol}}$
6   **end**
7   Choose $\vec{I}_{p_t}^{\text{m}} = max\{\vec{I}_{s_t}^{\text{m,sol}}, \ \forall \ s \in \{0, N_{\text{scales}}\text{-}1\}\}_{t=0}$
8   **for** $t \in \{0, N_{\text{Taylor}} - 1\}$ **do**
9   Update the model image : $\vec{I}_t^{\text{m}} = I_t^{\text{m}} + g \ [\vec{I}_p^{\text{shp}} \star \vec{I}_{p_t}^{\text{m}}]$
10   **end**
11   **for** $t \in \{0, N_{\text{Taylor}} - 1\}$ **do**
12   **for** $s \in \{0, N_{\text{scales}}\text{-}1\}$ **do**
13   Update the residual image : $\vec{I}_{s_t}^{\text{res}} = \vec{I}_{s_t}^{\text{res}} - g \ \sum_{q=0}^{N_{\text{Taylor}}-1}[\vec{I}_{sp_{tq}}^{\text{PSF}} \star \vec{I}_{p_t}^{\text{m}}]$
14   **end**
15   **end**
16 **until** *Peak residual in* $\vec{I}_{0_0}^{\text{res}} < f_{\text{limit}}$

```python
for i in range(niter):

    # Lines 2 - 6
    mscale, mx, my, mval = find_global_optimum(hsmmpsf,
            ihsmmpsf, smresidual, windowstack, findpeak)
    scale_counts[mscale] += 1
    scale_flux[mscale] += mval[0]

    # Line 7
    peak = numpy.max(numpy.fabs(mval))
    lhs, rhs = overlapIndices(ldirty[0, ...],  psf[0, ...], mx, my)

    # Lines 8 - 10
    m_model = update_moment_model(m_model,
            pscalestack, lhs, rhs, gain, mscale, mval)

    # Lines 11 - 15
    smresidual = update_scale_moment_residual(smresidual,
        ssmmpsf, lhs, rhs, gain, mscale, mval)

    if peak < absolutethresh:
        break
```

Figure 1: Minor cycle in python

4 Scale of problem

To study the scale of the problem without running large numbers of simulations, we have adapted the SKA Performance Model to ASKAP processing. The PM is a python library and accompanying Jupyter notebooks. The various steps in SKA processing have been expressed using the symbolic math capabilities of sympy. This allows both evaluation of a particular configuration of processing as a function of a range of parameters. We have developed a branch of the code specially adapted to ASKAP processing.

The SKA ICAL pipeline performs continuum imaging and self-calibration. A typical case for ASKAP is shown in figure 2.
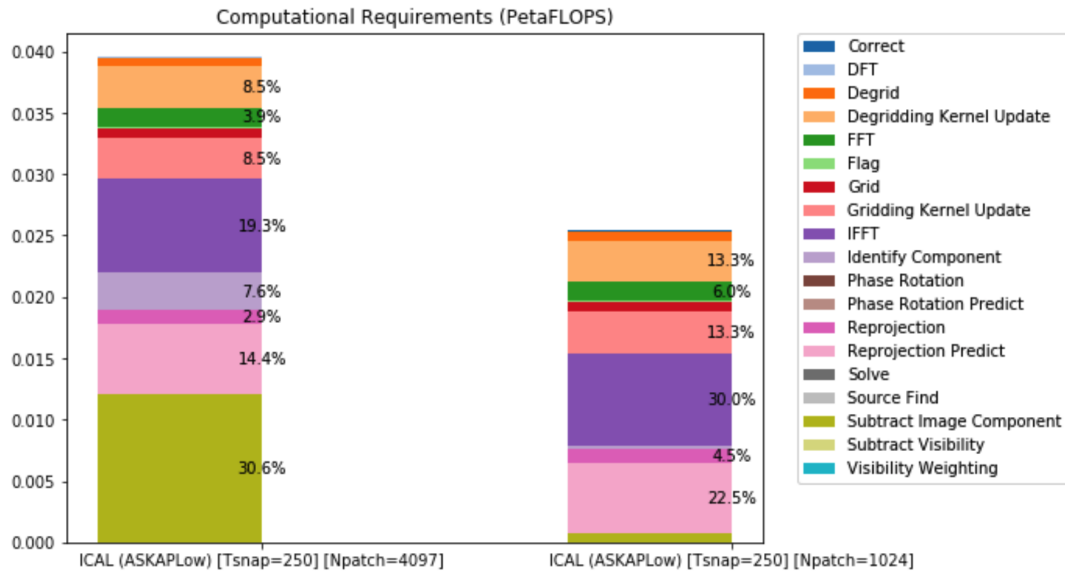
Figure 2: Estimation of the cost of processing steps from the SKA Performance Model as adapted to ASKAP. Left column is for large PSF patch size, right is for patch size of 512, which is the recommended value for ASKAP processing.

Identify Component in the SKA model corresponds to line 2 - 6 in 1, and Subtract Image Component to lines 7 - 15. The other parts of the processing in the histogram do distribute with reasonably high efficiency.

The minor cycle is currently purely serial and therefore constitutes a loss of processing efficiency in a parallel processing context. For N processors, and with a fraction p being amenable to parallelisation, Amdahl's law constrains the maximum speedup to be:

$$S = \frac{1}{(1-p) + \frac{p}{N}} \tag{1}$$

The resulting curves are shown in figure 3.

The fraction of processing, p, for M&M is about 4 %. Thus the maximum possible speedup of the ICAL pipeline is about 25.

Document no.: TCC-CASS-001   Unrestricted
Revision: 0.9   Author: T.J. Cornwell
Release date: Monday 18th June, 2018 09:51   Page 13 of 38
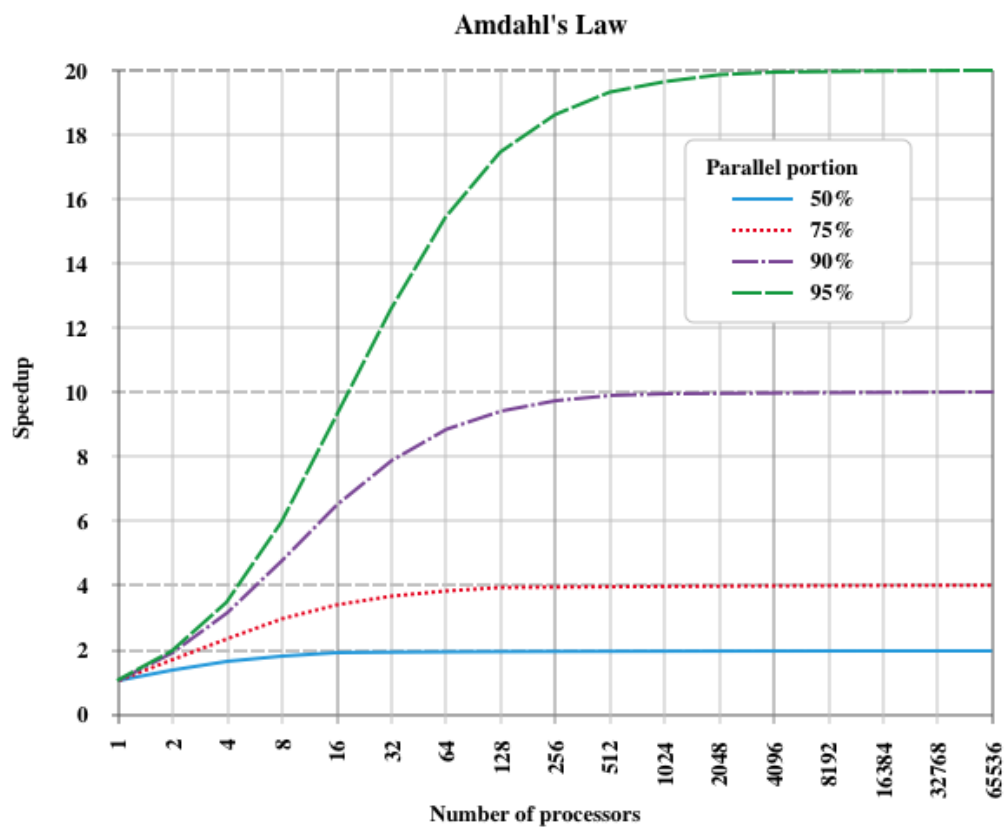
Figure 3: Amdahl's law for various fractions of parallel portion.

Document no.: TCC-CASS-001

Revision: 0.9

Release date: Monday 18th June, 2018 09:51

Unrestricted

Author: T.J. Cornwell

Page 14 of 38

5 Distribution

In this section, we discuss the opportunities for distribution of the inner loop over multiple processing nodes.

The axes of the inner loop are (RA, Dec, Scale, Moment). The numerical ranges of these parameters are as follows:

RA, Dec The image sizes for ASKAP are typically 4k x 4k or less. For SKA, numbers are much larger, up to 100k x 100k.

Scale Typically 3 - 5 scales are used, in a roughly geometric progression such as [0, 3, 10, 30, 100].

Moment Between 3 and 5 (frequency) moments are typical.

Distributing over scale will at most gain a factor of order 5. The network costs will be very significant.

More opportunity for distribution lies in the spatial axes. Spatially the coupling occurs for two reasons: the point spread function and the MS blobs. The point spread function can usually be assumed to be sufficiently well-behaved that updating of the residuals is only need in a limited neighbourhood. This statement is scale dependent and larger scales will inevitably show more coupling. It will be necessary to deal with components close to the partition boundary. Thus we will require enforcing some continuity constraint at the partition boundaries. There are a number of ways of doing so. We consider each of these in order of increasing complexity:

S1. Partition into sub-images Divide the image space into a number of disjoint partitions and clean each separately. The drawback to this approach is that large scales close to the boundaries will be penalised, leading to discontinuities in the final image.

S2. Overlap the partitions To avoid the penalisation of large scales in S1, each partition could be extended with a suitably sized guard band. At the end of the minor cycle, neighbours could be smoothly interpolated.

S3. Message passing inside the minor loop To avoid conflicts between components found in the overlap region, we could allow the neighbours to communicate every component found. We could partition as in S2 and communicate components found during the minor cycles, either peer-to-peer or to a central broker such as the master. Overlap by at least the largest scale size is required.

Each worker can check to see if a component overlaps with another worker or set of workers. If so, the component can be sent peer-to-peer or via the master. In theory MPI offers mechanisms for such communication but some experimentation might be necessary.

A message would be a flux value[scale, moment], a direction, and a shape, plus some ancillary information such as peak residual, taking O(100) bytes. For scaling, the communication of update events must occur in the same or less time as the worker execution times. The rate of overlap events is proportional to the fraction of the total image space that is subject to overlaps.

Document no.: TCC-CASS-001   Unrestricted
Revision: 0.9   Author: T.J. Cornwell
Release date: Monday 18th June, 2018 09:51   Page 15 of 38

S1 and S2 are relatively minor changes and can be implemented and tested with a moderate amount of effort. S3 is considerably more complex to implement and might be subject of the vagaries of MPI.

5.1 Data movement for S1, S2

In ASKAPsoft, at the end of the major cycle the different frequency residual images are sent to the master and combined into the residual frequency cubes. In the current single-threaded approach in ASKAPsoft, the residual images (per Taylor term) and PSF are then deconvolved on the master. Thus there are two synchronisation points: one to assemble the Taylor term cubes and one to await the end of the deconvolution.

In options S1 and S2, the simplest change would be to still combine the image planes on the master but to extract the patch residual images and PSF and distribute those to the workers. Each worker would then apply M&M to it's patch and return the resulting deconvolved image. The entire data flow to the master would be:

$$R_{IO,to\ master} = N_{\text{pix}} * N_{\text{chan}} \qquad (2)$$

Alternatively, instead of the data flow being centralised on the master, the workers could either broadcast the entire image of their frequency term to all workers or each worker could send just the appropriate patch of frequency to each worker.

$$R_{IO,pair-wise} = N_{\text{pix}}/N_{\text{Facet}} * N_{\text{chan}} * N_{\text{Facet}} \qquad (3)$$

Thus the data flow is the same in both cases but is bottlenecked to the master in the first approach.

The other important factor to consider is the working set size in both cases. In the *to master* approach the memory required in the master is significantly larger because the entire image cube has to be assembled before distribution to the workers

5.2 Overlaps and edge effects

S1 and S2 inevitably create seams in the resulting images where the deconvolution has been performed from different pixels either side of the seam. Making the seam narrow is unlikely to be satisfactory since even for the Hogbom clean the discontinuity will lead to different deconvolution either side of the seam. For the MultiScale clean, the same effect will occur at least on all scales in the deconvolution. Thus we can expect that the taper should be extend over more pixels than the largest scale.

Some possible tapering functions are shown in Figure 4. The simplest is simple addition or averaging, here shown as 'flat'. We improve by using a linear interpolation but this shifts discontinuities into the first derivative. The well-known Tukey filter removes these discontinuities by apply a cosine-taper, thus ensuring that only the second derivative is substantial. The computational cost in using the Tukey filter is negligible so we recommend always using it in preference to the flat of linear tapers.

The loss of efficiency due to the overlaps is non-negligible:

$$\epsilon_{overlap} = \left(\left(N_{pixel} - N_{facets} N_{overlap} \right) / N_{pixel} \right)^2 \qquad (4)$$

For a typical ASKAP case, we would need about 2048 pixels on a side, 8 facets on a side and an overlap of 32, say. For this example, the efficiency due to overlap is $(2048 - 256)/2048 = 0.875$. Increasing the overlap by a factor of two to 64, the efficiency is 0.75. However, we can expect to gain much more than this factor by distribution so we regard this level of inefficiency as acceptable.
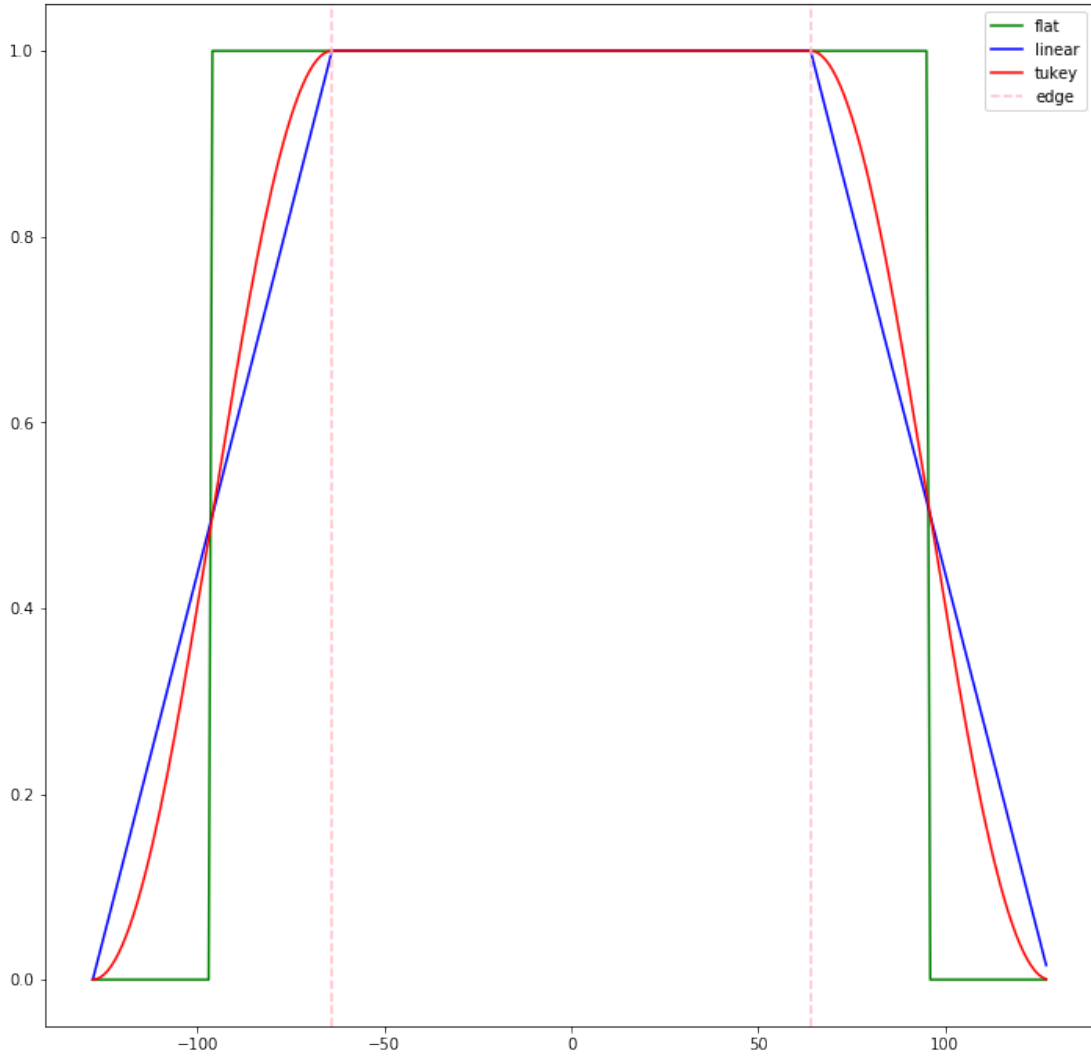
Figure 4: Tapers for feathering facet images. The vertical pink lines show the where the overlaps start. This is for 256 pixels and overlap of 64. For this case, the efficiency is roughly $(64/256)^2 = 1/16$

6 Simulations

The Algorithm Reference Library[2] was used for investigations. The ARL distributes imaging processing using the Dask library[3]. Since the ARL is python-based, the absolute processing times are not to be taken as realistic for ASKAPsoft, but we use only the relative times in our analysis.

We made simulations as follows:

- ASKAP antennas in 2km configuration,

- 11 frequencies spread from 0.85GHz to 1.15GHz, enabling fitting of Taylor series up to 5 terms (quintic). The bandwidth is sufficiently broad that over the frequency range far out sources can appear and disappear and then appear again at different frequencies. Hence 5 terms are used for many investigations.

- The source catalogs generated by querying the S3-SEX database[4], with a flux cutoff at 10uJy/beam and 100uJy/beam. The spectral index was derived from the 1400MHz and 610MHz fluxes.[5]

- Integrations spaced at 12 minutes, no noise added,

- Field center at ra=+30deg, dec=-60deg,

- Primary beam of 12m antenna with 2m blockage, no feed-legs.

- For some simulations the w term was set to zero to concentrate on the cleaning timing.

- The subimages are calculated so that the size is constant. Thus if the overlap is non-zero, the actual number of sub-images on an axis is two less than the nominal *e.g.* If the nominal is 8 then the actual is 6 and then there will be 36 sub-images spread over the two dimensions. We chose this approach to avoid having to analyze the behaviour for different size subimages.

Figure 5 shows a typical restored image obtained by running 5 major cycles M&M with 5 Taylor terms. The residual image (6) shows emission out into the first lobe of the primary beam.

---

[2]https://github.com/SKA-ScienceDataProcessor/algorithm-reference-library

[3]http://dask.pydata.org

[4]http://s-cubed.physics.ox.ac.uk/s3_sex

[5]Caveat emptor: S3-SEX contains a 730 Jy source in the catalog. We have removed this prior to these simulations on the basis that it should be dealt with by peeling instead of cleaning.
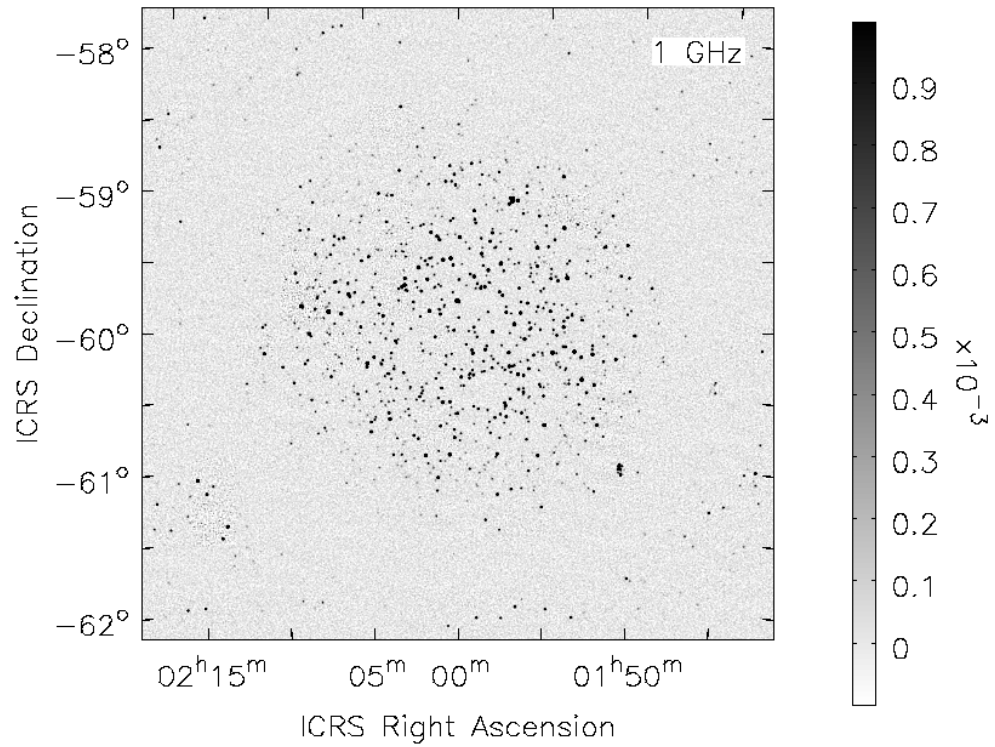
Figure 5: Test field imaged with the MSMFS, major/minor cycle algorithm in the ARL. The fit is to the first five Taylor terms (i.e. quintic). 16 facets on each axis were used, with overlap 16 pixels.
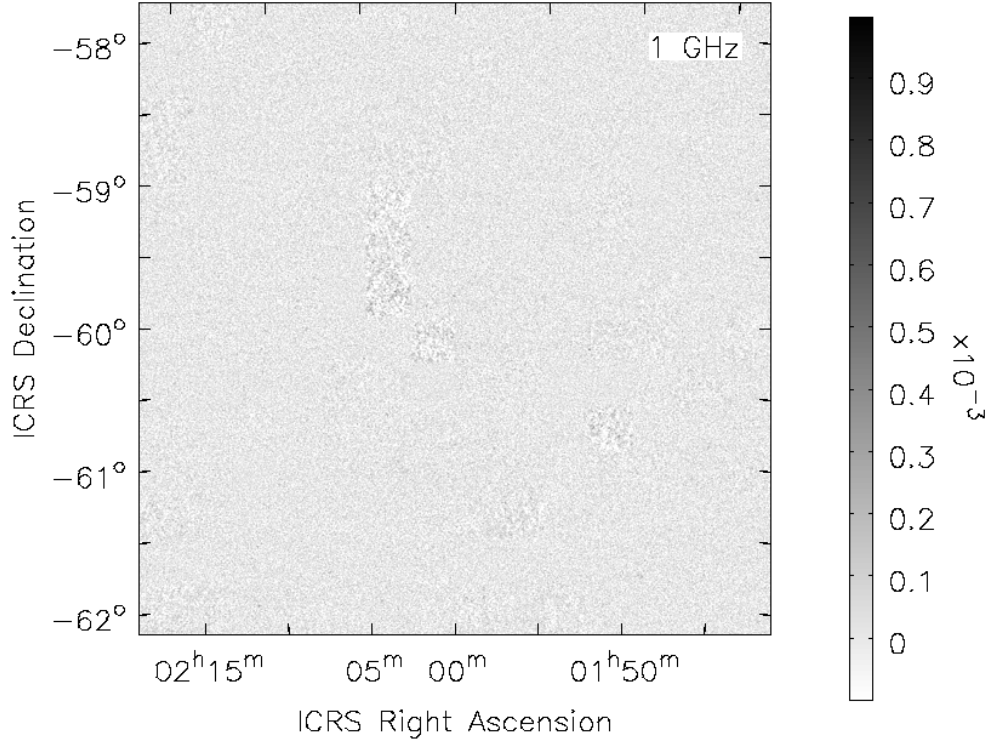
Figure 6: Residual for test field imaged with the MSMFS, major/minor cycle algorithm in the ARL. The fit is to the first five Taylor terms (i.e. quintic polynomial). 16 facets on each axis were used, with overlap 16 pixels.

A typical result is shown in figures 5 (restored image) and 6 (residual image).

6.1 PSF support

The first possible stumbling block is that the cleaning requires a PSF that crosses facet boundaries. If this was actually correct then the sub-image M&M would not be feasible.

In the current single threaded M&M the minor cycles, the processing consists of a search step, scaling as the number of pixels, and a subtraction step, scaling as the square of the size of the PSF support. The value of the PSF support used thus directly affects the processing time. Furthermore, the processing via sub-image inevitably means that a limit PSF support will be used.

For the simulated case, the RMS side-lobes of the (uniform weighted) synthesized beam are about 0.3% or less for pixels more than a few beams from the PSF peak. Hence we have reason to be optimistic that small values of PSF support will be acceptable given that we use a major/minor cycle. It is therefore important to determine the relationship is between final image quality and the PSF support.

We investigated the impact of limit PSF support using the simulations described above. In figure 7, we show the fractional error between the images obtained with the restricted PSF and the full PSF as a function of the support. The curve shows that the the effects of limited PSF support are much less than the criterion to switch to a major cycle (typically 10% or less).

The corresponding run times (wall clock) are shown in figure 9. The scaling is as expected: for small supports, the time is roughly independent of the PSF support, whereas for larger values, the scaling steepens as the component subtraction comes to dominated.

Clearly, in the context of a major/minor cycle algorithm, small PSF support can be used: the speedup is significant and there is no significant increase in error compared to the cutoff for the transition from minor to major cycle. This is confirmed by evaluating the difference between images processed using M&M and Sub-image M&M after ten major cycles (see figure 8).

We should emphasize that this result also holds for the current use of the single threaded M&M : the psf support can be set to a value much smaller than 512. One possible limit would be where the subtraction step becomes comparable to the search cost. This is obviously context dependent but we would expect values in the range 32 - 128. However, figure 9 shows that the gains are relatively small.
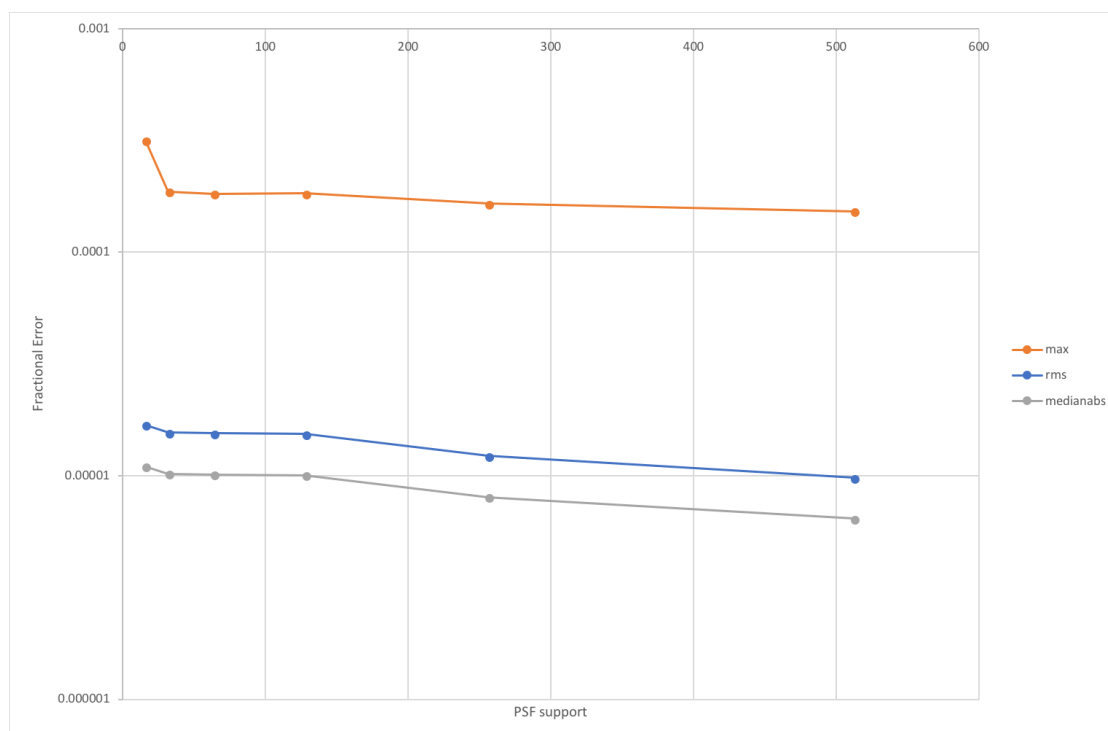
Figure 7: Fractional discrepancy as a function of PSF support. This shows a parallel 2048 by 2048 pixel, 3 term M&M clean in ARL. The computer was running 4 Dask workers with 32GB each. The discrepancy is the difference between the image processed with PSF support and the image processed with PSF support set to cover the entire image.
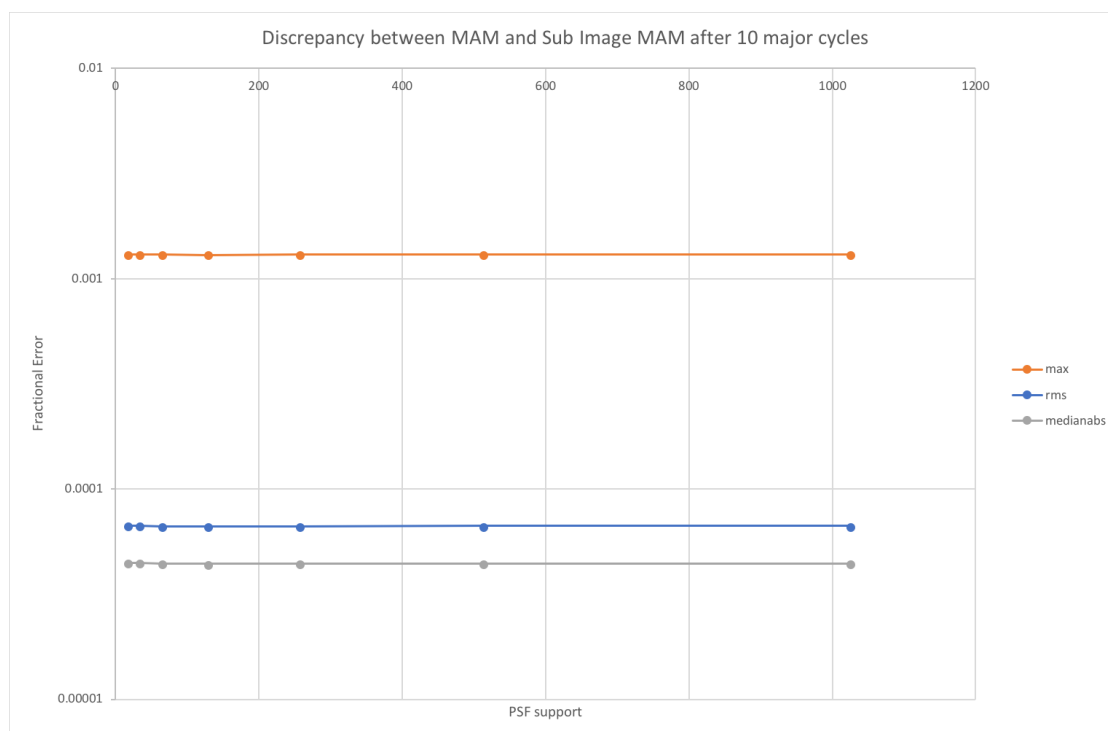
Figure 8: Fractional discrepancy as a function of PSF support after five major cycles. This shows a parallel 2048 by 2048 pixel, 3 term M&M clean in ARL. The computer was running 4 Dask workers with 32GB each. The discrepancy is the difference between the image processed with PSF support and the image processed with PSF support set to cover the entire image.
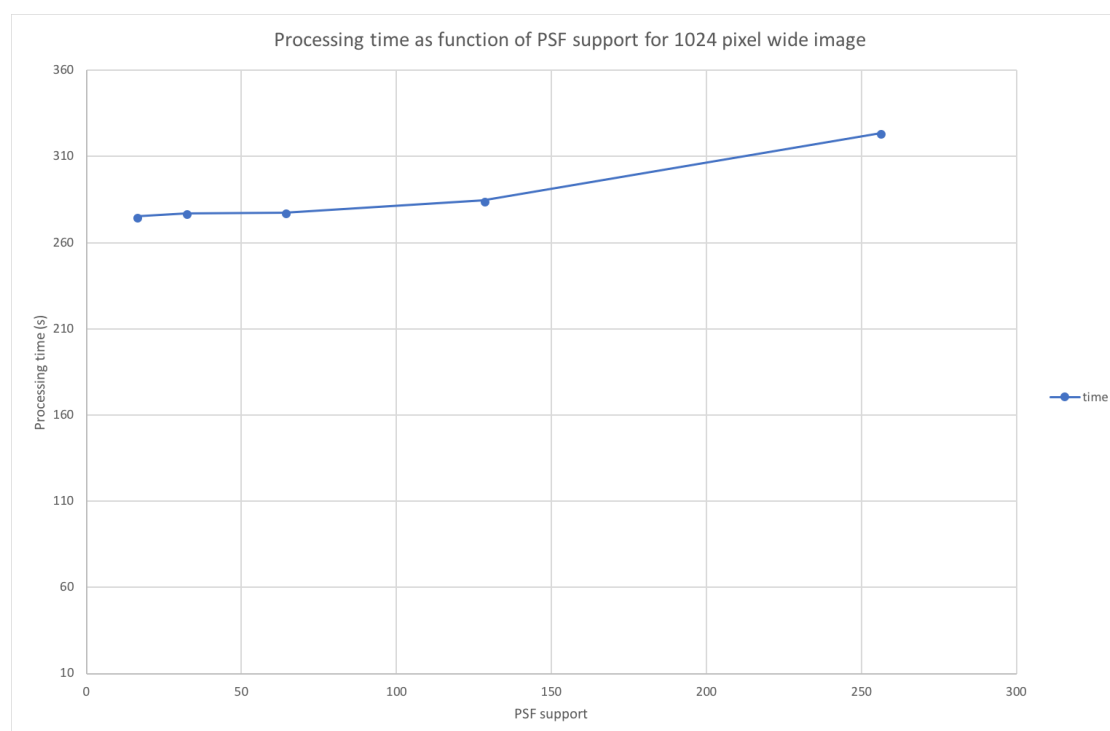
Figure 9: Processing time (seconds) as a function of PSF support. This shows a parallel 1024 by 1024 pixel, 3 term M&M clean in ARL. The computer was running 4 Dask workers with 32GB each.

6.2 Number of facets, and overlap

We turn now to the connection between the partitioning of the image by the used of overlapped facets and the performance of the sub-imaging algorithm. Accepting the performance as given by the existing single-threaded code running in parallel on many sub-images, the key question is whether the processing load is well-balance and provides strong scaling.

The primary cause of load imbalance is that different sub-images will have different numbers of sources and hence different elapsed time for deconvolution. In the graph-based approach used in the ARL this is observed directly. In ARL it is somewhat mitigated by the intrinsic flexibility of the graph-based work allocation.

If the image is made with a guard band of one or two primary beams, the outermost sub-images will mostly have weak sources and require little deconvolution initially. For the example above of 8 by 8 sub-images, about 16 will have substantial flux to deconvolve. This means that at the beginning of a major/minor cycle algorithm, the maximum scaling is about 16, although we divided the image into 64 sub-images. Using smaller sub-images by a factor of two would bring this scaling limit up to about 64. Essentially the scaling is determined by the number of sub-images we can define across the primary beam.

We can improve on this rough argument by some straightforward modeling. To a reasonable approximation, the processing time for a given sub-image will be proportional to the fraction of

Document no.: TCC-CASS-001
Revision: 0.9
Release date: Monday 18th June, 2018 09:51

Unrestricted
Author: T.J. Cornwell
Page 24 of 38

a flux in that sub-image. For a given model of the sky, we can calculate the fraction of time taken for each sub-image. We illustrate that point in figures 10, 11, and 12. The first plot is a realistic astronomical field constructed from the S3-SEX catalog, selected for all sources brighter than 10uJy/beam. The curve shows the fraction of flux per sub-image against rank of that flux. The slope comes from the logN/logS statistics of the S3-SEX catalog. The peak value shows roughly the processing for the worst sub-image. This provides an upper bound on the scaling possible. The mean number of processors is an estimate derived by dividing the total amount of processing by the peak value needed.

The next plot (figure 11) shows the effect of the ASKAP (single) primary beam. Most sub-images do not require cleaning and all the processing goes to the brightest fields. As processing proceeds thorough a number of major cycles, the brightest sources are been removed and eventually there remains a sea of sources spread other the primary and into the primary beam side-lobes. For the final major cycles, we can model the flux as simply by clipping the image with the primary beam at some maximum brightness such as 1mJy. The resulting scaling shown in figure 12 is then better balanced.

So, to summarise, we expect the scaling to be poor for the initial major cycles (typically ~ 5 for 16 facets per axis) where the brightest sources in the main lobes are removed, but it will improve (typically to ~ 50) as the cleaning goes deeper. Since the number of weak sources is much greater we can expect the scaling to be close to that on figure 12.
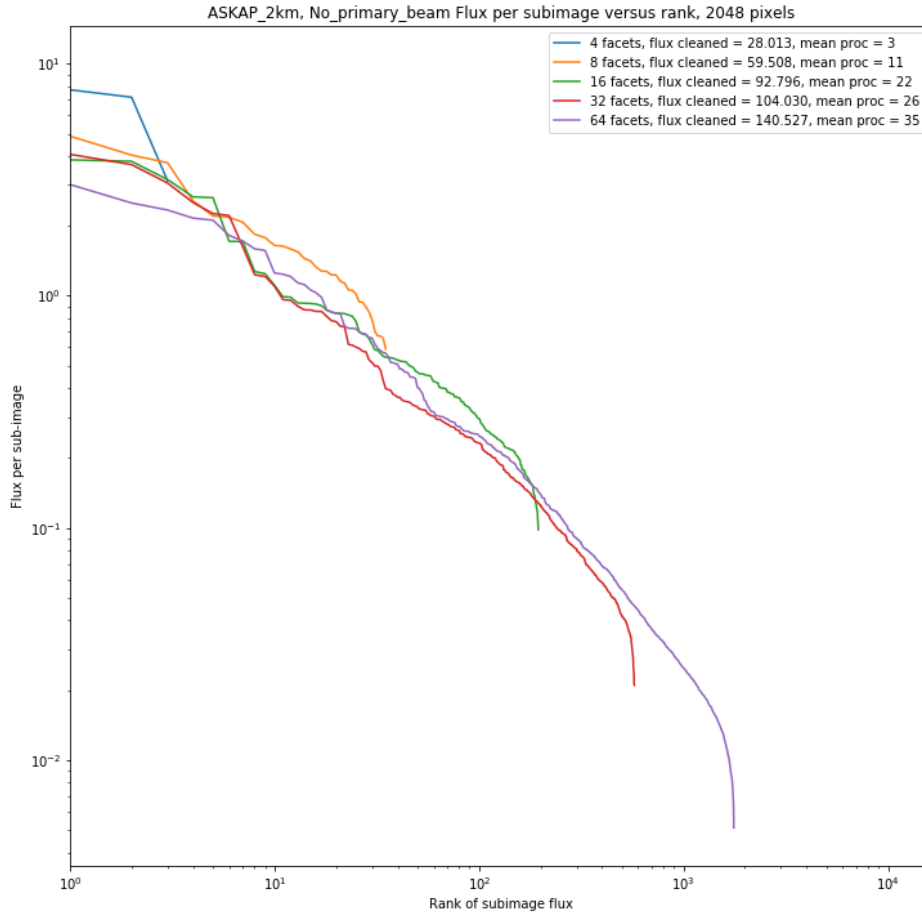
Figure 10: Integrated sub-image flux versus rank, for realistic image (constructed from S3-SEX) without a primary beam applied. Scales are [0, 3, 10], and the overlap is 16 pixels. The slope for all curves is mainly due to the statistics of image which is derived from radio source LogN/LogS relationship. This shows the complete absence of strong scaling i.e. adding more processors does not bring commensurate improvements in performance.
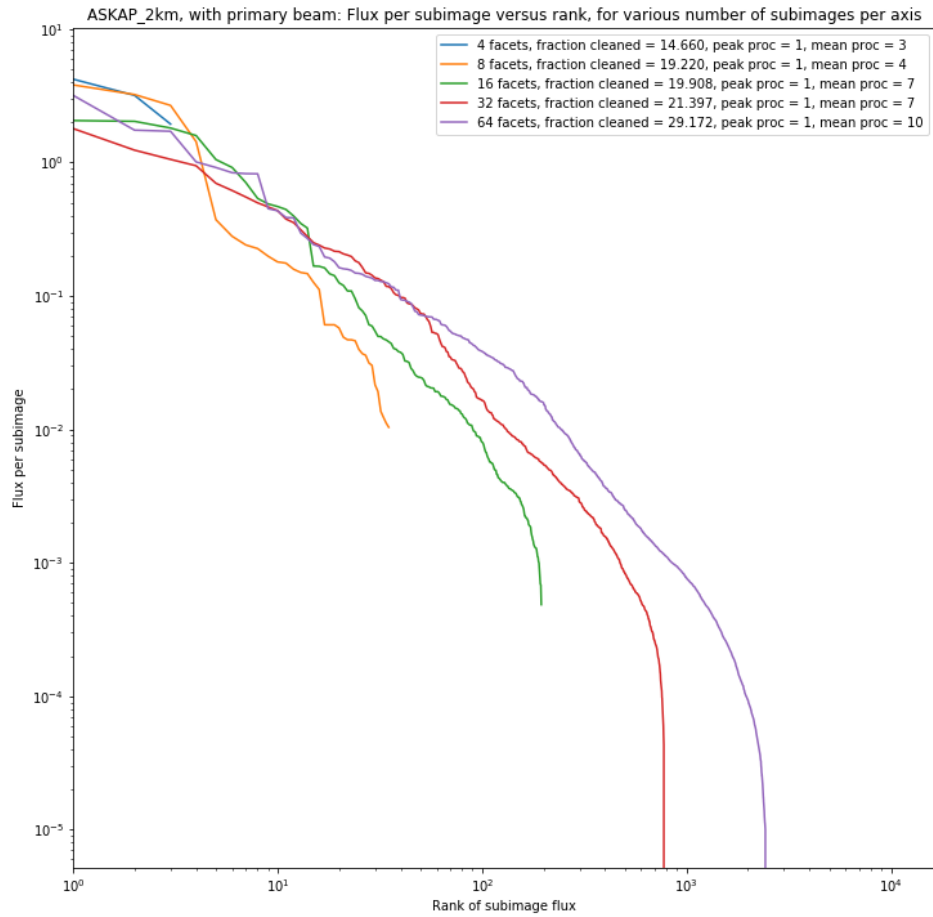
Figure 11: Integrated sub-image flux versus rank, for realistic image (constructed from S3-SEX) with a primary beam applied. Scales are [0, 3, 10], and the overlap is 16 pixels. Compared to figure 10, the primary beam application shifts the high rank sub-images lower in flux. The scaling has improved slightly.

Document no.: TCC-CASS-001
Revision: 0.9
Release date: Monday 18th June, 2018 09:51

Unrestricted
Author: T.J. Cornwell
Page 27 of 38
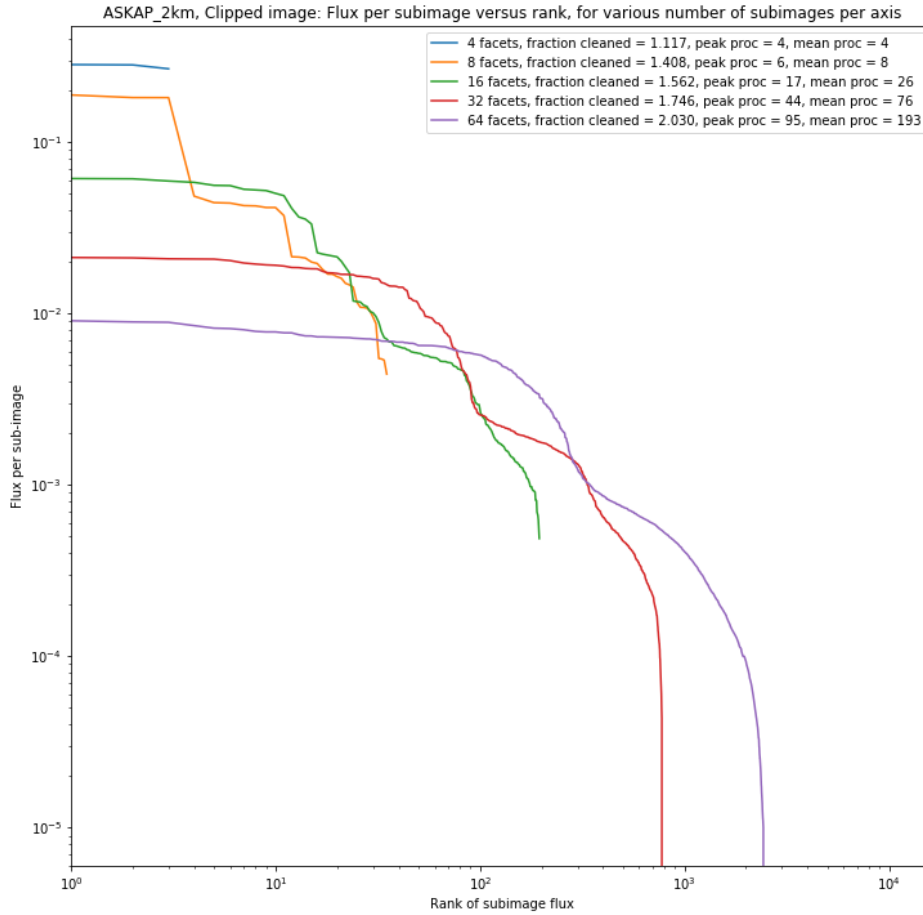
Figure 12: Integrated sub-image flux versus rank for various numbers of sub-images, for realistic image (constructed from S3-SEX) with a primary beam applied and then clipped at 100uJy/beam. Scales are [0, 3, 10], and the overlap is 16 pixels.

To get the overall scaling curve we must sum the work required for a sequence of major cycles going successively deeper. The thresholds start at $\sqrt{0.1}$ Jy and diminish by $\sqrt{10}$ as we proceed through 10 major cycles. We sum the results for each threshold. We perform this analysis for both 2km and 6km configurations. The results are shown in figure 13 and 14. The scaling is driven by the larger number of weak sources. The flatness of the curves as a function of rank means that strong scaling is possible for 200 - 300 processors. Adding more processors beyond this limit brings diminishing returns, and the overall efficiency drops.

Document no.: TCC-CASS-001
Revision: 0.9
Release date: Monday 18th June, 2018 09:51

Unrestricted
Author: T.J. Cornwell
Page 28 of 38
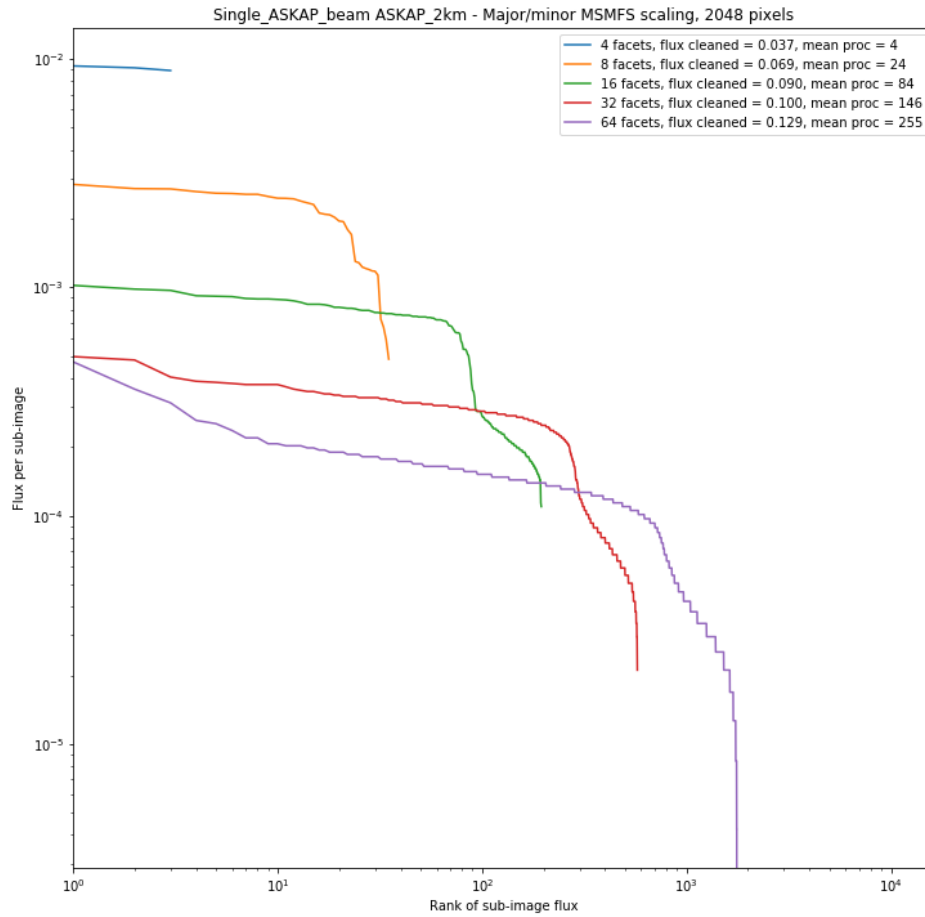
Figure 13: Aggregate scaling curve for 2km ASKAP configuration on multiple major cycles. Fraction of flux versus rank for various numbers of facets, for realistic image (constructed from S3-SEX) with a primary beam applied and then clipped at 1mJy/beam. Scales are [0, 3, 10], and the overlap is 16 pixels. The image size is 2048 by 2048 pixels. The scaling of work with numbers of sub-images (i.e. independent processors) is much closer to flat.

We have repeated this calculation for the full 36 antenna, 6km ASKAP (see figure 14).

Document no.: TCC-CASS-001   Unrestricted
Revision: 0.9   Author: T.J. Cornwell
Release date: Monday 18th June, 2018 09:51   Page 29 of 38
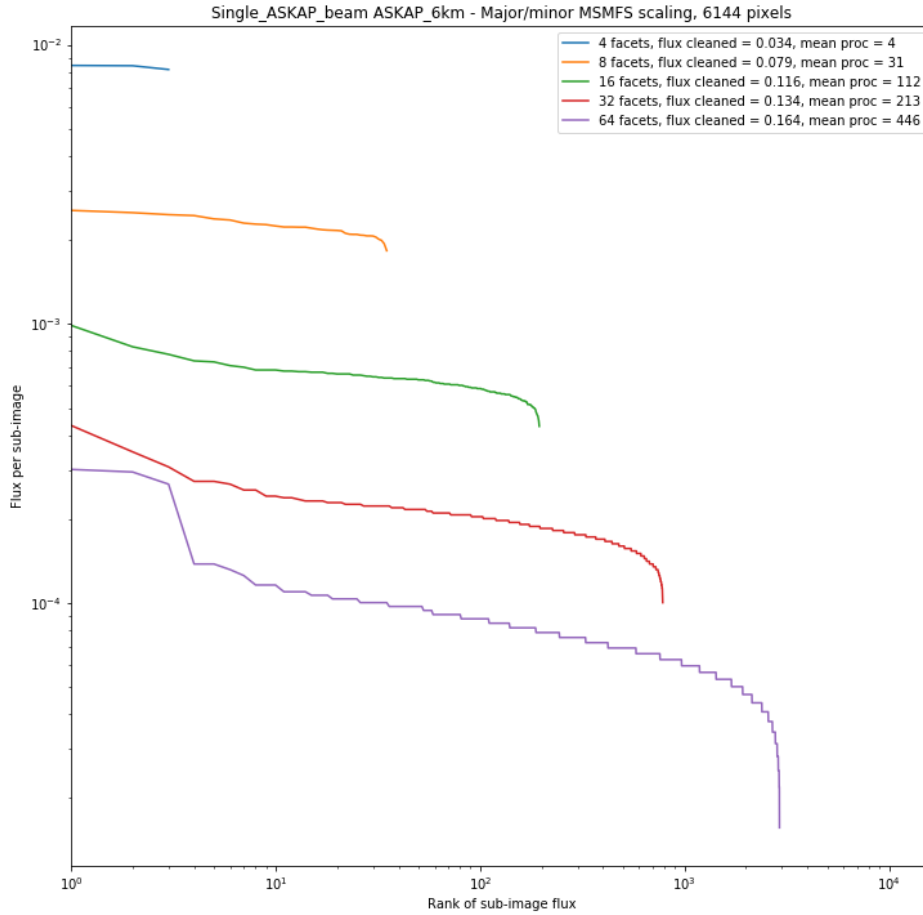
Figure 14: Aggregate predicted scaling curve for 6km configuration. Fraction of flux versus rank for various numbers of facets, for realistic image (constructed from S3-SEX) with a primary beam applied and then clipped at 1mJy/beam. Scales are [0, 3, 10, 30], and the overlap is 48 pixels. The image size is 6144 by 6144 pixels.

If the deconvolution were to be done on the mosaiced data then the full mosaiced primary beam (see figure 15) would be appropriate. This primary beam is larger and flatter so we would expect that the behaviour would be similar to the no-primary beam approach. The results for mosaiced primary beam are shown in figure 16. (Compare this to figure 13).
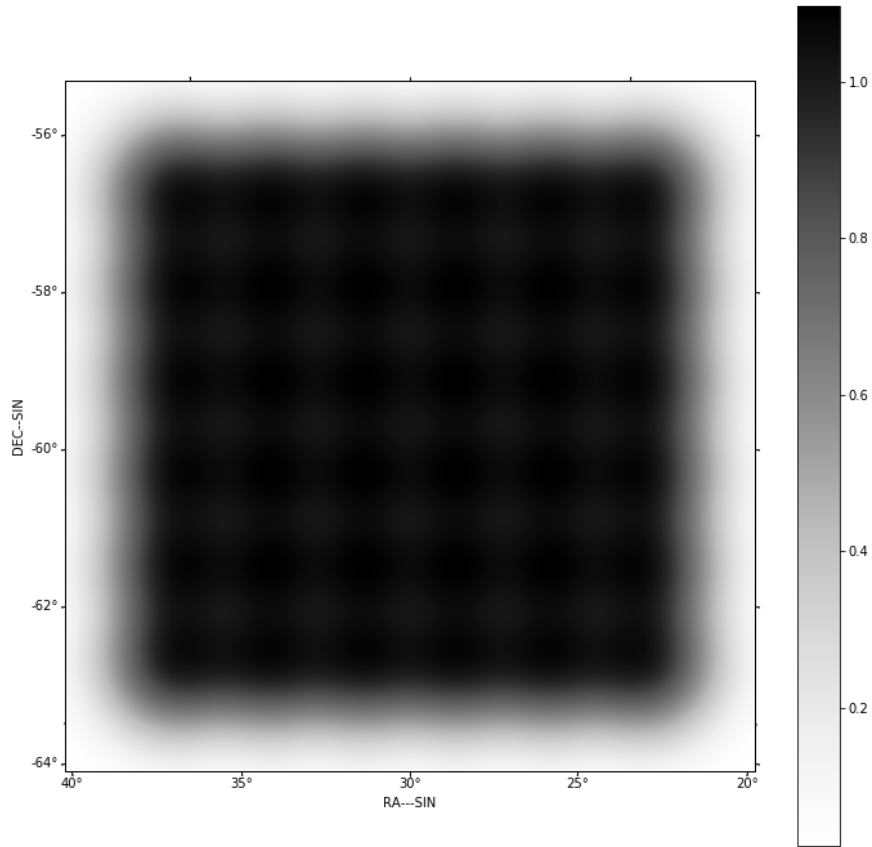
Figure 15: Full ASKAP primary formed by summing PAF beams in quadrature. The image size is 4096 by 4096 pixels.

Document no.: TCC-CASS-001   Unrestricted
Revision: 0.9   Author: T.J. Cornwell
Release date: Monday 18th June, 2018 09:51   Page 31 of 38
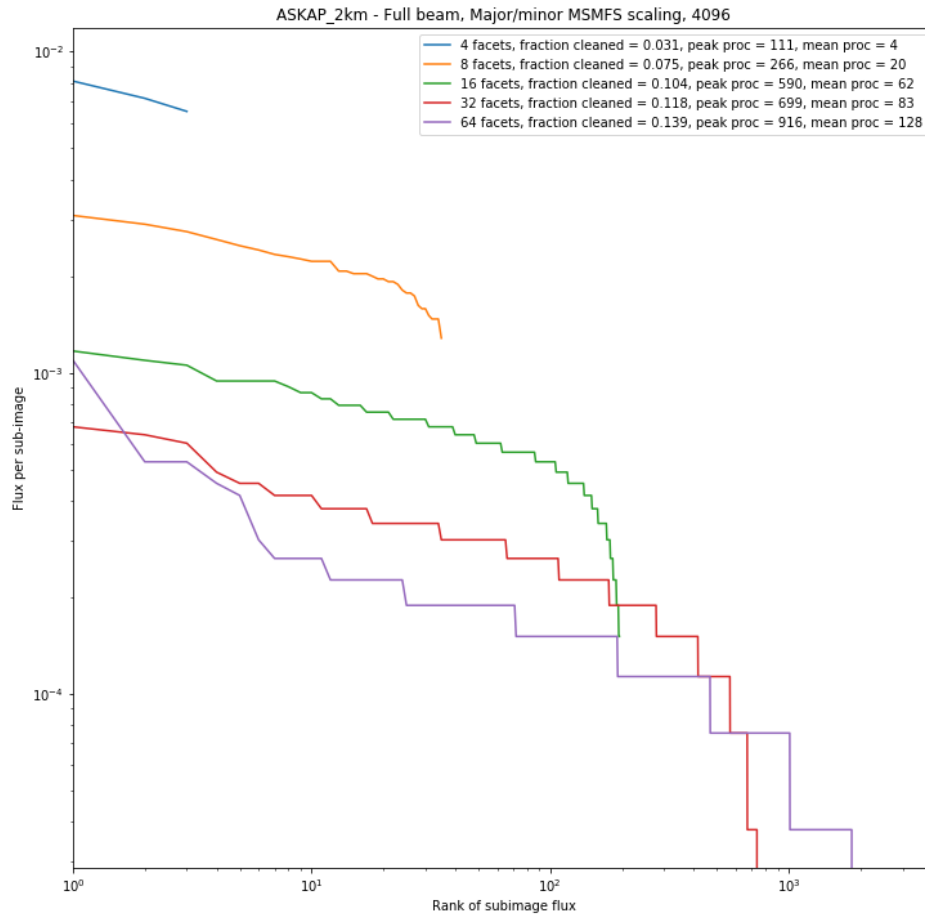
Figure 16: Aggregate scaling curve for 2km ASKAP configuration using the full mosaic primary beam. Fraction of flux versus rank for various numbers of facets, for realistic image (constructed from S3-SEX) with a primary beam applied and then clipped at 1mJy/beam. Scales are [0, 3, 10], and the overlap is 16 pixels. The image size is 4096 by 4096 pixels.

Figure 17 shows the scaling of number of active sub-images with increasing cycle number in major/minor cycles. The processing scaling improves with cycle number.
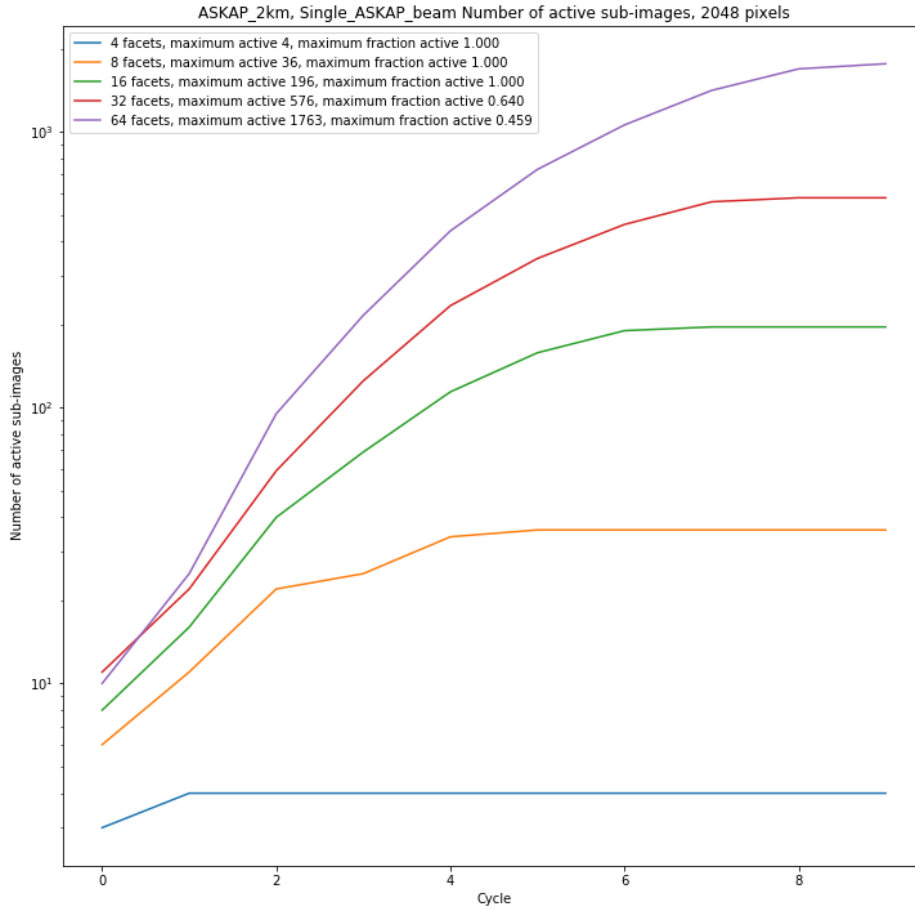
Figure 17: Number of active ranks vs major/minor cycle, for 2km ASKAP configuration using the single primary beam. The threshold for each cycle starts at $\sqrt{0.1}$ Jy/beam and proceeds in geometric progression to 0.1 mJy/beam.

6.3 Confirmation of scaling

Although the above behaviour does need to be verified on a large scale cluster, there is much that can be learned using small scale simulations on an 8 core Linux desktop.

In figure 18 we show time taken per cycle for a simulation of the 2km ASKAP configuration using the single primary beam. In figure 19 we show the processing time as a function of sub-image for each cycle.

Figure 18: Cumulative Measured time taken for the clean of each sub-image vs sub-image rank for each major/minor cycle, for 2km ASKAP configuration using the single primary beam. The threshold for each cycle starts at $\sqrt{0.1}$ Jy/beam and proceeds in geometric progression to 0.1 mJy/beam. The work increases steadily with cycle. Note that there is a stopping problem that can consume extra time, as well as the costs of a major cycle.

Document no.: TCC-CASS-001

Revision: 0.9

Release date: Monday 18th June, 2018 09:51

Unrestricted

Author: T.J. Cornwell

Page 34 of 38

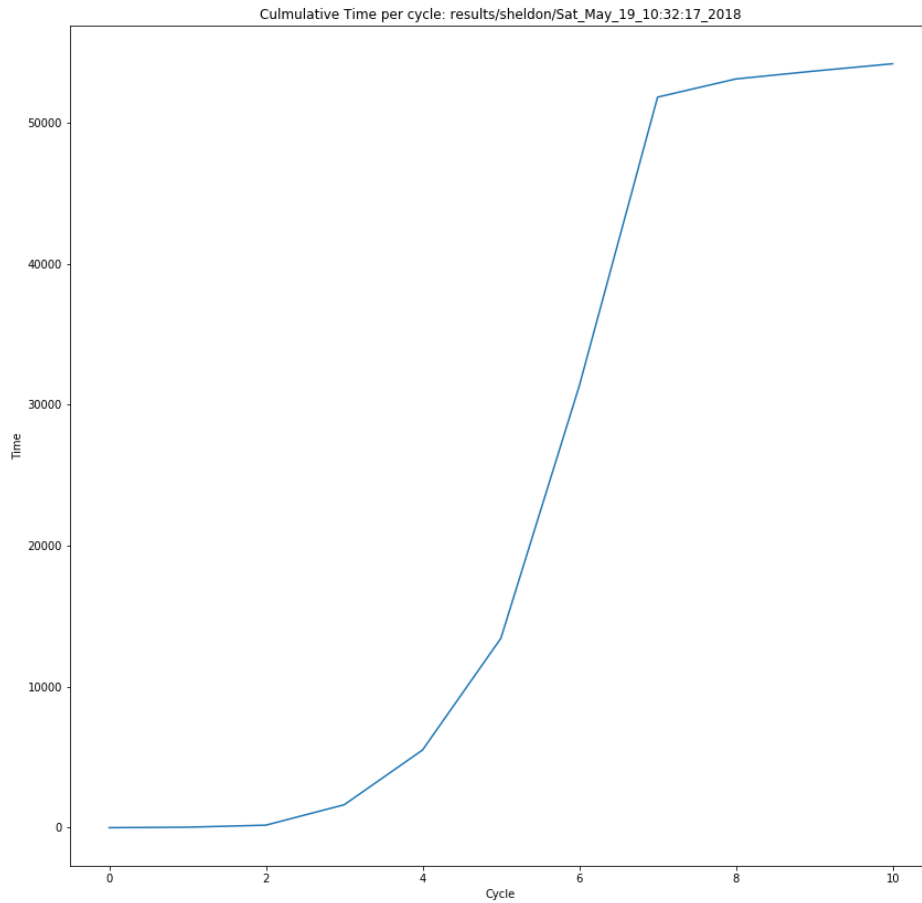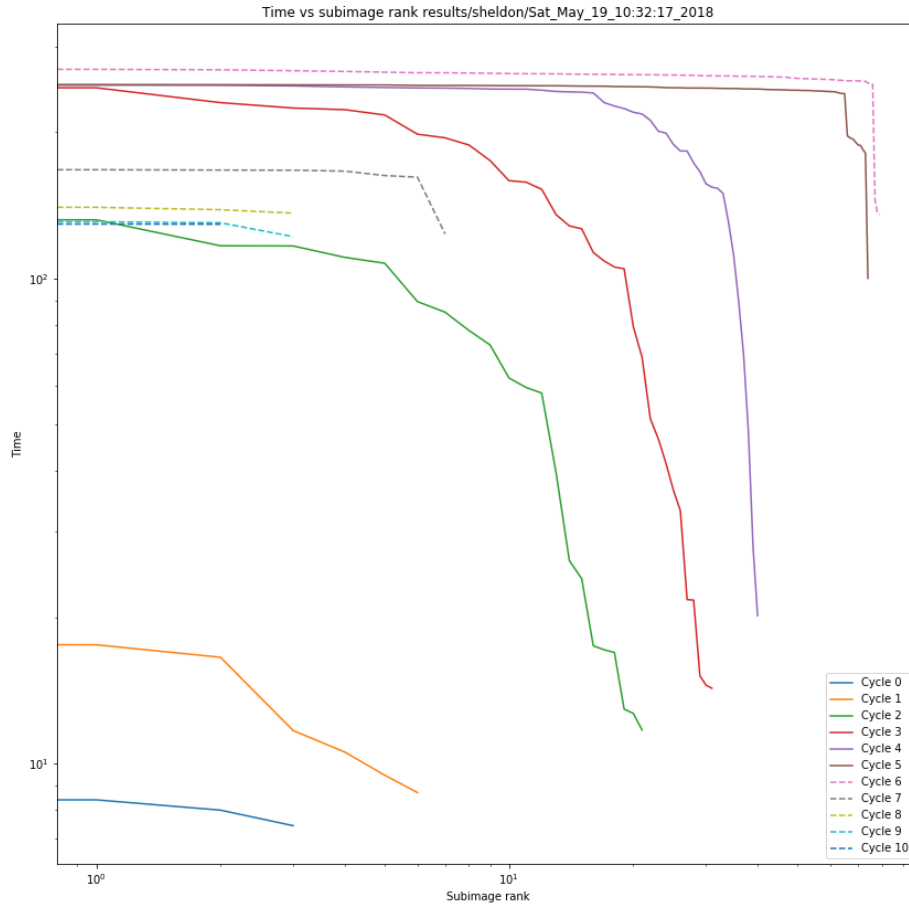**Figure 19:** Measured time taken for the clean of each sub-image vs sub-image rank for each major/minor cycle, for 2km ASKAP configuration using the single primary beam. The threshold for each cycle starts at $\sqrt{0.1}$ Jy/beam and proceeds in geometric progression to 0.1 mJy/beam. The work increases steadily with cycle. Note that there is a stopping problem that can consume extra time, as well as the costs of a major cycle.

Document no.: TCC-CASS-001   Unrestricted
Revision: 0.9   Author: T.J. Cornwell
Release date: Monday 18th June, 2018 09:51   Page 35 of 38

7 Porting to ASKAPsoft

The sub-image M&M algorithm is now present in the ARL and can be run on both simulated and real data. Porting the algorithm to ASKAPsoft requires implementation in C++. ASKAPsoft has much of the necessary library capabilities, with the exception of sub-image scatter/gather.

The core ASKAPsoft M&M algorithm, DeconvolverMultiTermBasisFunction.tcc, can be retained as is.

For distributed processing, ASKAPsoft uses a master/worker pattern implemented using MPI. In the imager, the master apportions calculation of the normal equations (essentially the residual images) to the workers, gathers and sums the results, and performs a full image single threaded deconvolution using MSMFS. The updated model is then broadcast to all workers, and processing repeats with the new model. Different frequency channels are assigned to different workers. Forming and summing the normal equations is a reduction across the entire set of workers.

For the deconvolution step, it may be convenient to use a separate MPI work group. The ASKAPsoft master/worker framework nominally supports this.

Since the scaling curves are relatively flat (see figures 13 and 14), most processors will be kept close to fully busy while the most demanding (rank 0) case will be processing. Load balancing would then only be needed for the high rank tail and should be performed automatically by the master/worker pattern.

The major python-based ARL capabilities that would need to be reproduced in ASKAPsoft are for scatter and gather of an image into sub-images, allowing overlap and tapering.

At the high numbers of processors, reduction in the scatter/gather steps may be needed. The total amount of network traffic is the full image size times the number of Taylor terms. Before beginning the sub-image cleans, the sub-images have to be scattered to the workers.

The gather/scatter step could be implemented either centrally or via neighbourhoods of workers. The latter avoids the use of a large amount of memory on the master to store the entire image.

To facilitate development, the sub-image M&M should first be implemented in cdeconvolver to resolve the issues around use of the master-worker scheme and then ported to cimager. The implementation should accommodate both central and localised gather/scatter as described in the previous paragraph.

Document no.: TCC-CASS-001   Unrestricted
Revision: 0.9   Author: T.J. Cornwell
Release date: Monday 18th June, 2018 09:51   Page 36 of 38

8 Work plan

The work plan to bring the Sub-image M&M to ASKAPsoft. is straightforward. The steps required are:

- Develop sub-image scatter/gather in ASKAPsoft.

- Develop master/worker version of scatter/gather

- Deploy in cdeconvolver as a test harness

- Perform scaling tests and adjust as necessary

- Deploy in cimager

- Check for strong scaling

- Run on simulated and real ASKAP and SKA data

- Investigate scaling to SKA

- Write report

As part of this work, the stopping problem should be pursued further since this is leading to substantial overhead in unnecessary major cycles.

Access to ASKAP computing facilities will be necessary.

9 Conclusions

1. We have analysed the performance of the current single threaded M&M algorithm in ASKAPSoft. As if there were any doubt, according to the SKA Parametric Model as adapted to the full scale ASKAP, the processing can consume anything in the range 30% - close to 100%.

2. The motivation for a distributed algorithm is therefore clear. We have developed and demonstrated a distributed M&M algorithm that tiles the image space into partially overlapping sub-images and runs the single-threaded M&M on each sub-image. The sub-images are then appropriately combined back into a single final image cube, using a Tukey tapering function to reconcile the overlap regions. We call this algorithm the Sub-image M&M .

3. Sub-image M&M has been implemented in the SKA Algorithm Reference Library. The distribution of the processing is performed using a graph-based approach.

4. Although we have not had access to a cluster with large numbers of processors, we have performed an analysis of the large N scaling of Sub-image M&M . The scaling is ultimately determined by the number of sub-images that can be used without distorting the multi-scale deconvolution. For the ASKAP 2km configuration using scales of 0, 3, 10 pixels, this translated to about 32 - 64 sub-images (per axis). For the initial deconvolution cycles, the processing is limited to a few bright sub-images and the strong scaling is limited to 3 - 10 processors. However, as the algorithm proceeds deeper, the sources to be deconvolved occupy more of the image and so more sub-images are active, allowing strong scaling up to 30 - 100 processors. On aggregate, the algorithm shows strong scaling up to 50 - 100 processors (32 - 64 sub-images per axis with 16 pixel overlap).

5. For the 6km configuration strong scaling is predicted for up to 100 to 300 processors.

6. There will still need to be some work distribution to deal with the tail in the scaling curves. However, this should fit naturally into the master/worker paradigm used in ASKAPsoft.

7. This algorithm does not preclude the use of parallelization, such as OpenACC or CUDA to speed up the single sub-image MSMFS. As the speed of the M&M processing of a single sub-image improves, the overall speed should improve.

8. Finally, it is worth emphasizing that this approach works because the PSF of ASKAP (for the simulated) data is really good with rms sidelobes about 0.3%. Nearly all of the removal of the PSF sidelobes is actually performed via the major cycles.

9. The Sub-image M&M is not well suited for the case where many scales are needed to represent the structure. In that case it may be necessary to perform successive processing with increasing maximum uv baseline.

10. The predicted scaling has yet to be confirmed on a configuration with many (hundreds) nodes.

Document no.: TCC-CASS-001
Revision: 0.9
Release date: Monday 18th June, 2018 09:51

Unrestricted
Author: T.J. Cornwell
Page 38 of 38